uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Informática

2017-2018

Trabajo Fin de Grado

"Control de un robot con Raspberry Pi"

Eduardo Miguel García Romero

Tutor

Ángel García Olaya

Leganés, 2018

RESUMEN

La tecnología relacionada con la robótica es cada día más interesante. Los avances en esta rama son más grandes según avanza el tiempo, siendo cada vez más útiles los robots que sirven de ayuda para el ser humano. Robots como los de tipo industrial o domésticos cada vez poseen más funcionalidades que hacen que la sociedad avance hacía un mundo más automatizado.

En este caso, se trata el tema de los robots móviles, los cuales están diseñados con el objetivo de poder moverse en diferentes entornos. Habitualmente, el control de estos robots se realiza mediante órdenes automatizadas, con el objetivo de dar una tarea al robot y que éste la realice. Sin embargo, este uso no le da la posibilidad al usuario de manejar el robot de forma manual, lo cual les resta muchas posibilidades de uso a estos robots.

El presente Trabajo de Fin de Grado basa su desarrollo en la elaboración de un sistema completo para llevar a cabo un manejo manual de cualquier robot móvil compatible con la librería de control robótico ARIA. Para ello, se ha creado una aplicación móvil que, mediante acciones simples del usuario, sea capaz de comunicarse con el robot y ejecutar los diferentes movimientos disponibles.

Además, el sistema ha sido probado enteramente en la Universidad Carlos III de Madrid, con el objetivo de demostrar el correcto funcionamiento del sistema tanto en un entorno emulado como en un entorno real.

Palabras clave: Acciones, ARIA, Control, Móvil, Robot

DEDICATORIA

A mi tutor, Ángel, por confiar en mi para llevar a cabo este Trabajo de Fin de Grado, dándome cuando ha hecho falta.

A mis padres, Eduardo y Pilar, por toda la comprensión y apoyo que me han dado en todo momento y por los ánimos para continuar.

A mi pareja, Jéssica, por ayudarme a aguantar todo el peso del trabajo de estos años en la universidad y apoyarme en todo momento.

Y, por último, a todos mis compañeros del grupo UC3M Pals. Diego, José Manuel, Sergio y Manuel, muchísimas gracias por toda la ayuda y apoyo en estos años.

TABLA DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1. INTRODUCCIÓN Y OBJETIVOS

En la actualidad, el sector de los robots está muy extendido. A lo largo de los últimos años, empresas de robots móviles, como MobileRobots [5], han fabricado diferentes tipos de robots con diversos objetivos, como pueden ser la docencia, llevar a cabo trabajos sobre terrenos irregulares, facilitar la interacción del usuario con los robots o mejorar la eficiencia de diferentes tareas. En este caso, este Trabajo de Fin de Grado se centra en los robots móviles.

Los robots móviles tienen como característica principal la capacidad de moverse en su entorno, al contrario que los robots fijos industriales. Estos robots son un punto importante de la investigación actual de las universidades con laboratorios centrados en robótica.

El objetivo principal de este Trabajo de Fin de Grado es conseguir manejar de manera inalámbrica uno de los robots, permitiendo que éste sea capaz de realizar diferentes movimientos, como movimientos con distancias o giros con unos grados concretos.

1.1. Contexto

En la Universidad Carlos III de Madrid [2], el Departamento de Informática dispone de varias unidades del robot móvil Pioneer P3-DX. Este robot ha sido fabricado por Pioneer, en colaboración con MobileRobots [5], y dispone inicialmente de dos ruedas, cada una con un motor independiente, produciendo que el robot sea capaz de girar sobre su eje, permitiendo que las ruedas giren a diferentes velocidades y en diferentes direcciones. También dispone de un sonar frontal para conseguir un mejor control del entorno. Además, este robot es fácilmente personalizable, ya que existen decenas de accesorios probados por Pioneer para la utilización con este robot, como pueden ser láseres de control de distancias, cámaras de visión o brazos mecánicos para coger objetos.

Fig. 1.1 Pioneer 3 DX

Para llevar a cabo el control de este robot, es necesario utilizar una computadora que, conectada al robot, sea capaz de enviarle información. Con este objetivo, se dispone de ARIA [1]. ARIA es una interfaz de programación de aplicaciones de control de robots orientada a objetos, compatible con los robots de MobileRobots. ARIA está desarrollado en C++, y ofrece un control de alto nivel sobre el robot, además de sobre los diferentes accesorios que se le pueden añadir. Además, ARIA está disponible tanto para Linux, Mac y Windows, haciéndola compatible con todos los sistemas operativos actuales.

1.2. Motivación

Actualmente, el control de estos robots se realiza mediante conexiones cliente-servidor entre un ordenador y la placa controladora del robot, o bien únicamente la placa controladora ejecuta acciones sobre el robot mediante toma de decisiones de forma autónoma.

El primer tipo de control se realiza mediante una conexión TCP entre el cliente y el servidor. Sin embargo, esta conexión, que se realiza mediante la librería ArNetworking, incluida en ARIA, está desarrollada en C++, dificultando el desarrollo de plataformas móviles que utilicen este tipo de conexión utilizando esta librería.

El segundo tipo de control se realiza únicamente en la placa conectada al robot mediante la toma de decisiones con inteligencia artificial. Sin embargo, este tipo de control no es útil de cara al control de robots manualmente.

En la actualidad, no existe ninguna forma de controlar este tipo de robots de manera manual mediante el uso de smartphones, tanto Android como iOS. Según el portal web

We Are Social [3] en su estudio *Digital in 2018* [4], existen más de 5.000 millones de líneas móviles activas en 2018. Tal como explican estos reportes, los smartphones tienen un porcentaje de penetración en la sociedad del 68%. Esto quiere decir que, para los usuarios de smartphones, la utilización de estos para realizar tareas supone una mayor comodidad y atracción frente al uso de otro tipo de aparatos, como los ordenadores. Por lo tanto, la elaboración de un sistema de control para este tipo de robots en smartphones puede ser muy útil. Este problema es la principal razón por la cual se ha llevado a cabo el presente TFG.

1.3. Objetivos

Habiendo analizado los problemas encontrados en un primer estudio, el objetivo de este TFG se presenta como el desarrollo de una aplicación para plataformas móviles que, mediante conexión TCP con una placa conectada al robot, sea capaz de llevar a cabo sobre este un control manual, además de dotarle de ciertas acciones útiles de movimiento.

Este objetivo principal se divide en diferentes subobjetivos, los cuales se resumen a continuación:

- Desarrollar un servidor ejecutable en la placa conectada al robot, el cual ejecute las acciones requeridas sobre éste.
- Desarrollar un cliente capaz de enviar las acciones requeridas al servidor.
- Desarrollar una aplicación móvil que implemente al cliente, con el objetivo de enviar estas acciones desde el terminal móvil.

Además, este proyecto tiene varias utilidades posteriores a su desarrollo:

- Facilitar a estudiantes o investigadores futuros un control simple sobre el robot.
- Facilitar la docencia sobre el control de estos robots, mostrando una forma sencilla y práctica de llevarlo a cabo.

1.4. Estructura del documento

Para finalizar el presente apartado, se resume la estructura que va a seguir el TFG:

El primer capítulo, Introducción y Objetivos, muestra el inicio del proyecto, explicando la situación actual alrededor de los robots móviles, así como las posibilidades de control sobre estos. Se desarrollan los problemas actuales y el objetivo del presente TFG.

El segundo capítulo, Estado del arte, desarrolla de forma más completa todo lo relacionado con las partes involucradas en la realización de este TFG, como son los diferentes robots móviles que se pueden utilizar o las diferentes placas para soportar el servidor.

En el tercer capítulo, Análisis del sistema, se realiza un completo análisis inicial de todo lo relacionado con el sistema desarrollado, como son los diferentes tipos de usuarios del sistema y los requisitos necesarios.

En el cuarto capítulo, Diseño de sistema, se muestran las diferentes fases de desarrollo de sistema, mostrando todas las decisiones que afectan al diseño del proyecto.

En el quinto capítulo, Pruebas del sistema, se desarrolla un plan de pruebas en el que se incluyen tanto casos de uso como las comprobaciones necesarias sobre el sistema.

En el sexto capítulo, Planificación del proyecto, se desarrollan todas las fases seguidas durante la elaboración del presente TFG.

El séptimo capítulo, Marco Regulador, se tratan los apartados de legislación y estándares técnicos relacionados con la elaboración del proyecto.

En el octavo capítulo, Entorno socioeconómico, se desarrolla un análisis que muestra un desglose de los costes del desarrollo del proyecto, junto con un análisis de los posibles beneficios que pueda aportar el proyecto.

El noveno capítulo, Conclusiones, muestra todas las conclusiones obtenidas tras la finalización del proyecto.

En el décimo capítulo, Control of a Robot with Raspberry Pi, se incluye un resumen en inglés del TFG.

Por último, se incluye un último capítulo, Bibliografía y Acrónimos, que contiene toda la bibliografía utilizada para obtener información durante el desarrollo del TFG.

2. ESTADO DEL ARTE

En el presente apartado se muestra cómo se encontraban los diferentes apartados relevantes para el desarrollo del proyecto antes de llevarlo a cabo, junto con las diferentes alternativas de cada apartado.

2.1. Robot móvil

En la actualidad existe una amplia gama de robots móviles con los que es posible trabajar. Sin embargo, las alternativas valoradas en el apartado siguiente del presente documento limitan esta lista de robots posibles a los compatibles con ARIA, debido a las conclusiones establecidas en el apartado 2.2.3. Estos robots son los siguientes:

- **AmigoBot**

Fig. 2.1 AmiboBot

AmigoBot es un robot de pequeño tamaño, diseñado con el objetivo de ser eficiente en tareas de educación e investigación. Está completamente ensamblado y dispone de dos ruedas, un buzzer y un sonar omnidireccional. Su microcontrolador funciona bajo el sistema operativo AmigoOS, y utiliza el SDK de Pioneer, que es un conjunto de aplicaciones y librerías que facilitan y aceleran el desarrollo de proyectos relacionados con la robótica.

Sus especificaciones son las siguientes:

Especificaciones	
Nombre	AmigoBot
Peso	3.6 kg

Especificaciones	
Velocidad máxima frontal	1 m/s
Radio de giro	16.5 cm
Tiempo de autonomía	2-3 horas
Carga útil operativa	0 kg
Terrenos transitables	• Azulejo • Baldosas • Pavimento

TABLA 2.1 ESPECIFICACIONES AMIGOBOT

Este robot puede ser manejado desde cualquier PC de forma inalámbrica o mediante conexión ethernet.

- **PeopleBot**

Fig. 2.2 PeopleBot

PeopleBot es una plataforma robótica desarrollada para prestar servicios al usuario mediante interfaz gráfica. Cuenta con un chasis elevado con diferentes puertos de entrada y salida, mediante los cuales poder conectar accesorios y sensores adicionales. Cuenta con sensores sonar tanto en la parte superior como inferior, además de sensores de rayos infrarrojos para detectar cualquier objeto entre la parte superior e inferior del robot. Su microcontrolador funciona bajo el sistema operativo ArcOS, y utiliza el SDK de Pioneer.

Sus especificaciones son las siguientes:

Especificaciones	
Nombre	PeopleBot
Peso	21 kg
Velocidad máxima frontal	0.8 m/s
Radio de giro	33 cm
Tiempo de autonomía	8 horas
Carga útil operativa	8 kg
Terrenos transitables	• Azulejo • Baldosas • Pavimento

TABLA 2.2 ESPECIFICACIONES PEOPLEBOT

Este robot puede ser utilizado para mapear el entorno en diferentes alturas, así como para reproducir sonidos o patrullar la zona en la que se encuentra.

- **Pioneer 3 DX/AT**

Este robot ligero y pequeño dispone de dos ruedas manejadas cada una por un motor independiente. Además, cuenta con sonar frontal. Está desarrollado principalmente para labores de enseñanza e investigación, su microcontrolador funciona bajo el sistema operativo ArcOS y utiliza el SDK de Pioneer.

Sus especificaciones son las siguientes:

Especificaciones	
Nombre	Pioneer 3 DX
Peso	9 kg
Velocidad máxima frontal	1.2 m/s

Especificaciones	
Radio de giro	26.7 cm
Máxima pendiente	25%
Tiempo de autonomía	8-10 horas
Carga útil operativa	17 kg
Terrenos transitables	• Azulejo • Baldosas • Pavimento

TABLA 2.3 ESPECIFICACIONES PIONEER 3 DX/AT

Este robot es el más versátil en cuanto a educación e investigación se refiere. Además, el Grupo de Planificación y Aprendizaje (PLG) de la Universidad Carlos III de Madrid [2] dispone de varios de estos robots, lo cual facilita la elección.

- **Pioneer LX**

Fig. 2.3 Pioneer LX

Este robot es una plataforma avanzada móvil programable a la que se le pueden añadir fácilmente accesorios y sensores. Incluye software de navegación y mapeo zonal, un joystick de manejo, sensores sonar frontales y laterales, y una computadora integrada, en la que puede instalarse tanto Windows como Linux. De esta forma, y al contrario que en el caso de los robots anteriores, no necesita un ordenador externo para controlarlo.

Sus especificaciones son las siguientes:

Especificaciones	
Nombre	Pioneer LX
Peso	60 kg
Velocidad máxima frontal	1.8 m/s
Radio de giro	34.3 cm
Máxima pendiente	20%
Tiempo de autonomía	13 horas
Carga útil operativa	60 kg
Terrenos transitables	• Azulejo • Baldosas • Pavimento

TABLA 2.4 ESPECIFICACIONES PIONEER LX

- **PowerBot**

Fig. 2.4 PowerBot

PowerBot es una plataforma robótica desarrollada principalmente para llevar a cabo tareas de investigación sin necesidad de parar largos periodos de tiempo para cargarse. Está preparado para que se le puedan instalar diferentes sensores, e incorpora una computadora integrada, sensores sonar y laser. Una de las características más interesantes de este robot es que cuenta con una base de carga independiente, a la cual puede acudir el robot cuando lo necesite, recargarse, y volver a sus tareas. Esto lo

convierte en una buena alternativa de cara a conseguir un robot que funcione de forma autónoma.

Sus especificaciones son las siguientes:

Especificaciones	
Nombre	PowerBot
Peso	120 kg
Velocidad máxima frontal	2.1 m/s
Radio de giro	54 cm
Máxima pendiente	15%
Tiempo de autonomía	4.5 horas
Carga útil operativa	75 kg
Terrenos transitables	• Azulejo • Baldosas • Pavimento

TABLA 2.5 ESPECIFICACIONES POWERBOT

- **Seekur**

Fig. 2.5 Seekur

Seekur es un robot de gran tamaño diseñado por Mobile Robots, desarrollado con el objetivo de desplazarse sin problemas a través de terrenos irregulares, siendo también resistente a diferentes climas. Todas sus ruedas funcionan de forma independiente y puede ser fácilmente modificable mediante la adición de diferentes accesorios. Su microcontrolador funciona bajo SeekurOS, y utiliza el SDK de Pioneer.

Sus especificaciones son las siguientes:

Especificaciones	
Nombre	Seekur
Peso	300 kg
Velocidad máxima frontal	1.8 m/s
Radio de giro	83 cm
Máxima pendiente	35%
Tiempo de autonomía	3-8 horas
Carga útil operativa	70 kg
Terrenos transitables	• Azulejo • Baldosas • Pavimento • Césped • Nieve ligera • Terreno rocoso • Arena

TABLA 2.6 ESPECIFICACIONES SEEKUR

Este robot, junto con su versión pequeña Seekur Jr., son los únicos robots fabricados por Mobile Robots que están pensados para atravesar terrenos irregulares, por lo que pueden ser muy útiles en tareas de investigación relacionadas con exploración de campo en la que no puede actuar un humano.

- **Seekur Jr.**

Fig. 2.6 Seekur Jr.

Seekur Jr. es un robot diseñado por Mobile Robots, basado en Seekur, desarrollado con el objetivo de desplazarse sin problemas a través de terrenos y climas irregulares. Al ser más pequeño que Seekur, se facilita el manejo de este robot, aunque ve su autonomía penalizada. Todas sus ruedas funcionan de forma independiente, y puede ser fácilmente modificable mediante la adición de diferentes accesorios. Su microcontrolador funciona bajo SeekurOS, y utiliza el SDK de Pioneer.

Sus especificaciones son las siguientes:

Especificaciones	
Nombre	Seekur Jr.
Peso	77 kg
Velocidad máxima frontal	1.2 m/s
Radio de giro	52 cm
Máxima pendiente	75%
Tiempo de autonomía	3-5 horas
Carga útil operativa	40 kg

Especificaciones	
Terrenos transitables	• Azulejo • Baldosas • Pavimento • Césped • Nieve ligera • Terreno rocoso • Arena

TABLA 2.7 ESPECIFICACIONES SEEKUR JR.

2.2.Control del robot: Servidor

Con el objetivo de ejercer un control sobre el robot, se han valorado dos alternativas: ARIA y ROS. En este apartado, se desarrolla una descripción de cada uno de los servicios y, finalmente, una conclusión explicando la elección de la alternativa utilizada.

2.2.1. ARIA

Como se ha mencionado en el apartado 1 del presente documento, ARIA [1] es la interfaz nativa de control de los robots de MobileRobots [5]. Es una interfaz de programación de aplicaciones de control de robots orientada a objetos. Ofrece una manipulación sobre el robot tanto a alto nivel como a bajo nivel. Está desarrollada en C++, pero ofrece dos *wrappers* para utilizar lenguaje Java o Python para el desarrollo tanto del servidor como del cliente. Estos *wrappers* son clases elaboradas en el lenguaje que se va a utilizar, las cuales tienen la capacidad de invocar los métodos del lenguaje original e interpretar los resultados obtenidos de sus funciones. Esta característica ha sido esencial en el desarrollo del proyecto, ya que el objetivo de hacer una aplicación móvil con la que controlar al robot no habría sido posible sin esto.

Además de los correspondientes *wrappers*, ARIA aporta varios ejemplos tanto para Java como para Python, así como muchos más ejemplos desarrollados en C++. Pese a tener los ejemplos escritos en Java, estos son muy simples, no mostrando bien el funcionamiento del robot. Para observarlo mejor, es necesario consultar tanto la documentación como los ejemplos hechos en C++, ya que estos son bastante más completos y aportan más.

Por otro lado, ARIA ofrece soporte para otras funcionalidades mediante la adición de otras librerías, como control de distancia por sonar o por láser.

En un primer momento, es posible probar los ejemplos aportados en ARIA sobre un simulador desarrollado por MobileRobots, denominado MobileSim [6]. Este software sirve como simulador de robots de MobileRobots y sus entornos, con el objetivo de desarrollar el software controlador del robot sin necesidad de tenerlo a mano. MobileSim utiliza datos de escenarios con obstáculos, los cuales son desarrollados mediante software externo como Mapper3 [8] o Mapper3Basic [9]. Estos entornos pueden ser desarrollados mediante el robot utilizando sensores laser, o bien manualmente.

Además, con el objetivo de monitorizar y controlar al robot de forma remota, MobileRobots ha desarrollado también MobileEyes [7], una aplicación gráfica que utiliza la librería ArNetworking para conectarse con el servidor. Dependiendo de las especificaciones aportadas por el servidor, MobileEyes es capaz de representar la posición del robot en el escenario.

Por último, MobileEyes ofrece controles para manejar al robot de forma remota, cambiar parámetros del robot, o enviarle una meta para que este se desplace.

2.2.2. ROS

El Sistema Operativo Robótico (ROS) [10], desarrollado por el Laboratorio de Inteligencia Artificial de Stanford, es un *framework* utilizado principalmente para desarrollar software para robots, y que ofrece la funcionalidad de un sistema operativo.

Una de las partes más interesantes de ROS es que, además del sistema operativo como tal, existe un conjunto de paquetes desarrollados por la comunidad, denominado *ros-pkg*. Estos paquetes implementan distintas funcionalidades, como localización, simulación o percepción del entorno.

Tras realizar un análisis de la documentación existente sobre ROS, la mayoría de los proyectos desarrollados con este *framework* se desarrollan en un alto nivel, lo cual podría complicar el proceso de desarrollo del proyecto en vista de que el objetivo es llevar a cabo un control simple sobre el robot.

2.2.3. Elección de la alternativa

Tras analizar ambas alternativas y debatir con el cliente acerca de cuál sería una mejor opción, se ha decidido utilizar ARIA como *framework* de control del robot gracias a su facilidad de uso, además de ofrecer *wrappers* para Java y Python, lo cual facilita bastante el desarrollo del servidor.

ROS, pese a tener un unas características muy completas y útiles gracias a su amplia comunidad, es una alternativa demasiado compleja para el desarrollo del presente proyecto, lo que ha hecho que esta alternativa sea desechada.

2.3. Control del robot: Cliente

En este apartado se desarrollan las diferentes alternativas para la creación del cliente que usará el usuario final para manejar al robot. Ya que este control se va a realizar en remoto, es necesario analizar alternativas para diferentes tipos de terminales que puedan ejecutar un cliente de este tipo. Así, se han estudiado las posibilidades de computadora y de terminales móviles. Tras este análisis, se evalúan todas las opciones, exponiendo las ventajas e inconvenientes de cada una de ellas y, finalmente, una conclusión acerca de la elección tomada.

2.3.1. Alternativas en computadora

Ya que es necesario una computadora para desarrollar el proyecto, está claro que una clara alternativa son las computadoras. Las posibilidades de utilizar tanto un ordenador de sobremesa a distancia, o un ordenador portátil junto al robot, son alternativas a valorar.

Para desarrollar el cliente en computadora, existen múltiples alternativas, de las cuales se van a analizar el uso de una interfaz de línea de comandos y la elaboración de una aplicación de escritorio

- **Interfaz de línea de comandos**

Una CLI (Interfaz de línea de comandos) es una consola o representación basada en texto en la que el usuario puede escribir ordenes en forma de texto para manejar las funcionalidades de un software específico. Es un método que, inicialmente, puede ser más complejo de ejecutar para un usuario nuevo, pero que permite explorar todas las posibilidades que el software aporta.

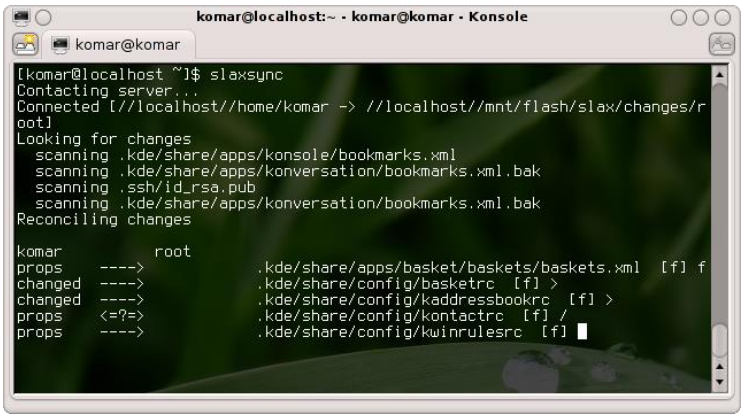

Fig. 2.7 Ejemplo CLI

Entre sus ventajas, encontramos la posibilidad de ejecutar todas las opciones del software ya que está enfocado en la funcionalidad; mayor velocidad de ejecución de las ordenes, ya que no es necesario moverse entre botones y pestañas gráficas; y, además, requiere de pocos recursos, ya que solo se maneja con texto y no es necesario visualizar ninguna imagen.

Del lado de las desventajas, una de las principales es la complejidad de uso. Lo habitual en un usuario normal es que no sepa qué hacer cuando se encuentra una interfaz de este tipo, haciendo que su aprendizaje de uso pueda ser más difícil. Además, los comandos también son complejos de aprender, por lo que puede dificultarse aún más. Por otro lado, la falta de una interfaz gráfica atractiva puede provocar que el usuario opte por no utilizar una CLI, haciendo inútil su desarrollo.

- **Aplicación de escritorio con interfaz gráfica**

Una GUI (Interfaz Gráfica de Usuario) es una representación gráfica mediante la cual el usuario puede interactuar con software o dispositivos mediante ciertos iconos. La mayoría de los usuarios utilizan a diario este tipo de interfaces, por lo que a priori, puede ser una buena alternativa de desarrollo.

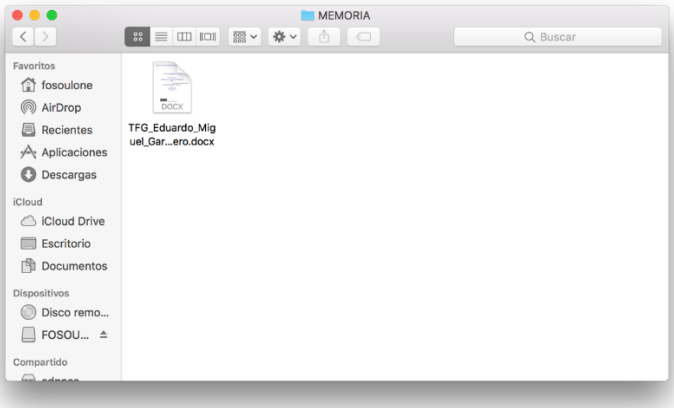

Fig. 2.8 Ejemplo GUI

Entre sus ventajas, una de las principales es que es más intuitivo que la CLI. La curva de aprendizaje es más sencilla y visualizar gráficamente las opciones a realizar hace más sencillo para los usuarios la utilización de estas interfaces. Además, este tipo de interfaces ofrece una forma muy sencilla de llevar a cabo multitarea, ya que con un solo clic puedes cambiar de pestaña o ventana sin dificultad. Por último, la posibilidad de representar gráficamente datos es una de las características principales de este tipo de interfaces, ya que las hace mucho más atractivas para los usuarios.

Por otro lado, entre sus desventajas se encuentran un uso bastante más alto de recursos del sistema, debido a la necesidad de representar diversos datos e imágenes en pantalla, y la falta de control que se tiene sobre el software. Esto es debido a que habitualmente no se pueden explorar todas las funcionalidades disponibles de software mediante una interfaz gráfica. Sin embargo, esto no afecta a la mayoría de los usuarios finales, ya que esta opción está más orientada a especialistas o expertos.

2.3.2. Alternativas en terminales móviles

Ya que el mercado de terminales móviles es de los más importantes actualmente en relación con la informática, estos terminales son la segunda opción por evaluar para desarrollar el cliente. Dentro de las opciones posibles para desarrollar la aplicación, se han evaluado las aplicaciones nativas, pseudonativas e híbridas.

- **Aplicaciones nativas**

Las aplicaciones nativas son aquellas que se desarrollan específicamente para un sistema operativo, siendo necesario adaptar el lenguaje de programación en caso de

querer cambiar de sistema operativo. Por lo tanto, teniendo iOS y Android como principales sistemas operativos móviles, hay que analizar las propiedades de cada uno. Además, una de las principales ventajas del desarrollo de las aplicaciones nativas es la existencia de IDEs muy avanzados en la actualidad, lo que hace más sencillo el desarrollo y la resolución de errores.

El sistema operativo de Apple, iOS, utiliza los lenguajes Objective C o Swift para programar sus aplicaciones, utilizando para ello el IDE XCode. Sin embargo, la publicación de una aplicación en la tienda de aplicaciones de Apple, Apple Store, requiere del pago de una licencia de desarrollador con coste anual.

Por otro lado, el sistema operativo de Google, Android, utiliza los lenguajes Java o Kotlin para el desarrollo de aplicaciones para esta plataforma. Se utiliza el IDE Android Studio y publicar la aplicación en la tienda de Google, Play Store, es gratuito.

Existen múltiples ventajas de este tipo de aplicaciones:

  - Las aplicaciones nativas permiten el uso de funciones avanzadas que las plataformas ofrecen a estas.
  - El rendimiento de estas aplicaciones es superior al resto de alternativas, utilizando su propia interfaz de usuario nativa.
  - Funcionan independientemente de la disponibilidad de conexión a internet en el dispositivo en caso de que la funcionalidad de la aplicación no la requiera.
  - Posibilidad de realizar trabajo en segundo plano.
  - Implementación de niveles de seguridad más completos y elevados que en el resto de las alternativas.
  - Mejoran la duración de la batería del terminal, ya que el código está optimizado para la arquitectura que utiliza el dispositivo.

- **Aplicaciones pseudonativas**

Las aplicaciones pseudonativas son aquellas en las que, mediante el uso de distintos *frameworks*, se genera el código nativo de cada plataforma a partir de código escrito en otros lenguajes, como Python, C# o JavaScript.

Realmente estas aplicaciones trabajan sobre los componentes nativos del terminal, por lo que están más cerca de las aplicaciones nativas que las híbridas en cuanto al nivel de limitaciones que se presentan.

Sus principales ventajas se centran en la similitud que tienen con las aplicaciones nativas, aunque ofrecen ciertas desventajas, como un rendimiento inferior al de las aplicaciones nativas, menor código común entre plataformas que en las aplicaciones híbridas, o una mayor dificultad de comprensión de las APIs nativas.

Los *frameworks* más comunes utilizados en función del lenguaje común que se utilice para desarrollar la aplicación son los siguientes:

  - Xamarin, para lenguaje C#, ofrece compatibilidad con Android, iOS y Windows Phone, además de permitir publicar las aplicaciones sin disponer de licencias de pago.
  - NativeScript, para el lenguaje JavaScript, soporta actualmente iOS y Android, y permite el acceso a APIs nativas desde el propio lenguaje.
  - Kivy, para el lenguaje Python, el cual está más orientado a interfaces gráficas utilizadas en juegos.

- **Aplicaciones híbridas**

Las aplicaciones híbridas son aplicaciones desarrolladas en HTML5 que se encapsulan en aplicaciones nativas. Utilizan el componente *WebView* para ejecutarse en el terminal en lugar de usar el navegador. Además, ofrecen la capacidad de conectar esta parte web con el terminal mediante JavaScript, ofreciendo una gran parte de las funcionalidades propias del terminal que el navegador no puede ofrecer, como, por ejemplo, notificaciones *push* o compras dentro de las aplicaciones.

Su característica principal es que gracias a la elaboración de *plugins*, tanto aportados por la plataforma como por la comunidad, es posible utilizar una gran cantidad de las funcionalidades nativas del SDK que ofrece el sistema del terminal. Sin embargo, esta característica está limitada por la propia existencia de estos *plugins*. Si se quiere utilizar una funcionalidad nativa para la cual no se ha desarrollado un *plugin*, será necesario desarrollar en el lenguaje nativo de cada plataforma. Además, el rendimiento sigue siendo inferior al de las aplicaciones nativas debido a la necesidad del uso de *WebView*.

Por otro lado, ciertos componentes nativos que utilizan componentes gráficos pueden ser difíciles de compaginar con *WebView*.

2.3.3. Elección de la alternativa para el cliente

Además de las características de las alternativas para el desarrollo de la aplicación, hay otros factores que hay que tratar. Al estar actualmente en auge el mercado de terminales móviles, habitualmente estos son más utilizados y accesibles que un ordenador. Además, debido a las autonomías de los robots y que se busca poder manejarlo a distancia, estar lejos del robot no es una posibilidad tan buena como poder utilizar un smartphone y poder manejar al robot de cerca. Por lo tanto, las opciones basadas en el desarrollo en computadoras se han desechado, por lo que solo es necesario elegir entre las aplicaciones móviles.

Dentro de estas opciones, también hay que tener en cuenta la disponibilidad de los tipos de plataformas en los que se puede desarrollar la aplicación. Las aplicaciones híbridas son una posibilidad que, en la actualidad, van cobrando fuerza e interés por parte de los desarrolladores. Sin embargo, con el objetivo de poder probar estas aplicaciones, es necesario disponer de terminales con los sistemas operativos previamente mencionados.

En este caso, la falta de disposición de un terminal iPhone con el sistema iOS, ha hecho que la decisión final pase a ser una aplicación nativa o una pseudonativa.

Teniendo en cuenta las ventajas que ofrecen las aplicaciones nativas en cuanto a rendimiento y disponibilidad de funcionalidades, y de la disposición de varios terminales con sistema operativo Android, se ha decidido que la alternativa para el desarrollo del cliente sea una aplicación nativa para dispositivos Android.

2.4. Placa controladora del robot

En el presente apartado se ha llevado a cabo un análisis sobre las diferentes alternativas de placas para ejecutar el servidor que se conecte al robot. Inicialmente, se consideró la idea de utilizar un ordenador portátil sobre el robot, debido a la eficiencia que pueden aportar en cuanto a potencia. Sin embargo, también se ha tenido en cuenta el alto precio que pueden llegar a alcanzar en el mercado a diferencia de otras alternativas de bajo coste y que disponen de la suficiente solvencia como para realizar un buen trabajo, como las computadoras de placas reducidas. Estas son computadoras completas en un

solo circuito, las cuales tienen todo lo que una placa base necesita para funcionar, como el microprocesador, los puertos de entrada/salida y la memoria RAM.

Dentro de este segmento de placas, a lo largo del tiempo se han desarrollado múltiples ejemplos, de los cuales se han analizado varias alternativas con el objetivo de escoger la computadora que mejor se ajuste a las necesidades del proyecto.

2.4.1. Raspberry Pi 3

La Raspberry Pi 3 es una computadora elaborada por la Fundación Raspberry Pi con el objetivo de ser empleada en la enseñanza en centros educativos. Su software es de código libre, y su sistema operativo oficial se llama Raspbian, basado en Debian. Sin embargo, acepta otros sistemas operativos basados en Linux, así como una versión de Windows 10.

Fig. 2.9 Placa Raspberry Pi 3 Model B

Esta placa ha tenido diferentes versiones desde su salida, siendo actualmente su última versión la Raspberry Pi 3 B. Sus especificaciones son las siguientes:

Especificaciones	
Nombre	Raspberry Pi 3 Model B
SoC	Broadcom BCM2837
CPU	1.4GHz 64-bit quad-core ARMv8
GPU	Broadcom VideoCore IV
Memoria RAM	1 GB

Especificaciones	
Almacenamiento	Ranura MicroSD
Puertos E/S	• 4 puertos USB 2.0 • Conector MIPI CSI • Conector RCA • HDMI • Jack 3.5 mm
Periféricos de bajo nivel	• 17 x GPIO • 1 x bus HAT ID
Conectividad de red	• Conector RJ-45 • Wifi 802.11n/ac • Bluetooth 4.2 BLE
Sistemas operativos soportados	• Debian (Raspbian) • Fedora (Pidora) • Arch Linux (Arch Linux ARM) • Slackware Linux • SUSE 65 Linux Enterprise Server for ARM • RISC OS2 • Windows 10
Precio	34,69 €
Consumo energético	800 mA

TABLA 2.8 ESPECIFICACIONES RASPBERRY PI 3 MODEL B

En base a sus especificaciones, la Raspberry Pi 3 es una placa bastante completa, con una buena compatibilidad de sistemas operativos, diferentes tipos de puertos E/S, y una buena conectividad de red. En adición a estas características, esta placa cuenta con una amplia comunidad de usuarios que aportan diferentes tipos de ayuda, tanto para el desarrollo para esta plataforma, como para la resolución de problemas mediante foros o correos electrónicos.

Esta alternativa es, a priori, la principal opción debido a su disponibilidad en el laboratorio de la Universidad Carlos III de Madrid [2].

2.4.2. ODROID-C2

El modelo C2 de ODROID, placa fabricada por la empresa HardKernel, salió al mercado para competir directamente con la Raspberry Pi 3. Cuenta con un procesador un poco más potente, además de doblar su memoria RAM.

Fig. 2.10 Placa ODROID-C2

Sin embargo, uno de los puntos en los que falla es en la conectividad. Esta placa no cuenta con tecnologías inalámbricas como Wifi o Bluetooth. Además, tiene mayor consumo energético que la Raspberry Pi 3.

Especificaciones	
Nombre	ODROID-C2
SoC	Amlogic S905
CPU	ARM Cortex®-A53(ARMv8) 1.5Ghz quad core
GPU	Mali-450 GPU
Memoria RAM	2 GB
Almacenamiento	Ranura MicroSD
Puertos E/S	• 4 puertos USB 2.0 • HDMI • 1 puerto Micro USB OTG
Periféricos de bajo nivel	• Puerto GPIO de 40 pines
Conectividad de red	• Conector RJ-54 • Receptor infrarrojo

Especificaciones	
Sistemas operativos soportados	• Ubuntu 16.04 • Android 6.0 Marshmallow
Precio	79,90 €
Consumo energético	0.5-2 A

TABLA 2.9 ESPECIFICACIONES ODROID-C2

2.4.3. ODROID-XU4

ODROID posee otros modelos de placas, como por ejemplo la XU4. Al igual que el modelo C2, posee componentes que la hacen más potente que la Raspberry Pi 3, sin embargo, se ve penalizada en la conectividad.

Fig. 2.11 Placa ODROID-XU4

Especificaciones	
Nombre	ODROID-XU4
SoC	Samsung Exynos5422 Octa
CPU	Cortex-A15 2Ghz y Cortex-A7 Octa core
GPU	Mali-T628 MP6
Memoria RAM	2 GB
Almacenamiento	• Ranura MicroSD • Modulo eMMC

Especificaciones	
Puertos E/S	• 2 puertos USB 3.0 • 1 puerto USB 2.0 • HDMI
Periféricos de bajo nivel	• Cabecera doble de 30 pines • Cabecera doble de 12 pines
Conectividad de red	• Conector RJ-45
Sistemas operativos soportados	• Ubuntu 16.04 • Android 7.1.1 LineageOS
Precio	54,90 €
Consumo energético	4 A

TABLA 2.10 ESPECIFICACIONES ODROID-XU4

2.4.4. OrangePi Plus 2

La placa OrangePi Plus 2 es una alternativa mejor que las placas ODROID en cuanto a conectividad. Además, mejora en especificaciones técnicas a la Raspberry Pi 3. Sin embargo, es menos eficiente energéticamente, además de tener una difícil disponibilidad de compra en internet.

Fig. 2.12 Placa OrangePi Plus 2

Especificaciones	
Nombre	OrangePi Plus 2
SoC	AllWinner H3
CPU	ARM Cortex-A7 quad-core
GPU	Mali-400 MP2
Memoria RAM	2 GB
Almacenamiento	• 16 GB eMMC Flash • Ranura MicroSD
Puertos E/S	• 4 puertos USB 2.0 • 1 puerto USB OTG • HDMI • Conector CSI • Jack 3.5 mm
Periféricos de bajo nivel	• Cabecera doble de 40 pines
Conectividad de red	• Conector RJ-45 • Receptor Infrarrojo • Wifi 802.11 b/g/n
Sistemas operativos soportados	• Android 4.4 • Ubuntu • Debian • Raspbian
Precio	66,50 €
Consumo energético	2 A

TABLA 2.11 ESPECIFICACIONES ORANGEPI PLUS 2

2.4.5. JaguarBoard One Plus

La placa de JaguarBoard es la única alternativa evaluada que utiliza un SoC Intel. Mejora en potencia a la Raspberry Pi 3, pero carece de conectividad inalámbrica.

Además, su precio es más del doble que el de la Raspberry y su disponibilidad pasa por su compra en tiendas asiáticas.

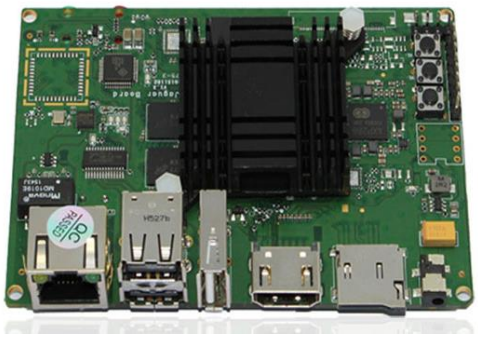

Fig. 2.13 Placa JaguarBoard One Plus

Especificaciones	
Nombre	JaguarBoard
SoC	Intel x86
CPU	Intel Atom Z3735F
GPU	Intel HD graphics 7Gen
Memoria RAM	2 GB
Almacenamiento	• 16 GB eMMC Flash • Ranura MicroSD
Puertos E/S	• 3 puertos USB 2.0 • HDMI • Conector Jack 3.5 mm
Periféricos de bajo nivel	• 4 pines GPIO
Conectividad de red	• Conector RJ-45
Sistemas operativos soportados	• Ubuntu • Debian • Linux Mint • Fedora Server • Windows 8.1-10

Especificaciones	
Precio	90,00 €
Consumo energético	2 A

TABLA 2.12 ESPECIFICACIONES JAGUARBOARD ONE PLUS

2.4.6. Elección de la placa controladora

Para la elección de la placa, se ha tenido muy en cuenta el factor de la conectividad, ya que el objetivo principal es poder conectar de forma inalámbrica un terminal a la placa. Con esta característica, según las especificaciones de las placas analizadas, únicamente la Raspberry Pi 3 y la OrangePi Plus 2 cuentan con conexión Wifi de forma estándar, pudiendo añadir en alguna de las otras alternativas algún accesorio capaz de brindar este tipo de conexión.

Por otro lado, hay que tener en cuenta el apartado de la eficiencia energética. Según el análisis de los robots, la autonomía de estos es algo escasa. Teniendo en cuenta que el robot que se va a utilizar es el Pioneer 3-DX, con una autonomía máxima de 10 horas, sería muy útil que la placa que se utilice consuma lo menos posible. En este caso, la propia Raspberry Pi 3 es la placa que tiene una mejor eficiencia energética debido a su bajo consumo eléctrico, por delante del resto de placas.

Por último, en relación con la potencia de las placas, es necesario destacar que la Raspberry Pi 3 es la placa menos potente de las analizadas, lo cual puede estar relacionado con el bajo consumo que esta requiere. En el presente proyecto, no es necesario que la placa utilizada sea la más potente, ya que únicamente se requiere que ejecute el servidor para controlar el robot.

Por lo tanto, en vista de que el Departamento de Informática de la Universidad Carlos III de Madrid [2] ya cuenta con unidades de la placa Raspberry Pi 3, y en base al análisis de las características de todas las alternativas, se ha decidido continuar el desarrollo del proyecto utilizando esta placa para controlar al robot Pioneer 3-DX.

3. ANÁLISIS DEL SISTEMA

En el presente apartado se desarrolla, teniendo en cuenta el Estado del arte, descrito en el apartado anterior, una especificación clara y concisa del sistema que se ha realizado.

Para llevar a cabo esta tarea, inicialmente se identifican a los usuarios potenciales del sistema, para poder un análisis de sus necesidades. Posteriormente, se ha realizado una definición del sistema, en la cual se exponen todas las funcionalidades que se han desarrollado, las cuales deben estar cubiertas por los requisitos de usuario y de sistema.

3.1. Identificación de usuarios

Tras estudiar las diferentes posibilidades que el sistema puede ofrecer a los usuarios, se han dividido a estos de la siguiente forma:

- Profesorado interesado en llevar el sistema a sus clases con el objetivo de la enseñanza.
- Investigador con el objetivo de estudiar la movilidad de estos robots, así como el funcionamiento del sistema y posibles mejoras.
- Usuario con objetivo de utilizar alguno de estos robots para llevar a cabo actividades de traslado de objetos pesados.

De estos grupos de usuarios, inicialmente los más potenciales son los dos primeros, ya que son estos dos grupos los que, por su trabajo en centros educativos, podrían disponer de alguna unidad de los robots de los puntos anteriores. Sin embargo, el tercer grupo de usuarios es también importante, ya que poder utilizar estos robots para desplazar objetos puede ser de gran ayuda tanto para personas que busquen transportar objetos pesados, como para personas con movilidad reducida que no tengan la capacidad de desplazar objetos pesados a ciertas distancias.

Haber dividido a los usuarios en estos grupos ha permitido poder realizar un buen análisis de funcionalidades necesarias para el sistema.

3.2. Definición del sistema

En este apartado se ha desarrollado, a partir del apartado 1.3 del presente documento, una definición clara y concisa de los objetivos y funcionalidades del sistema, con el objetivo de mostrar las características y capacidades del sistema.

Este sistema se basa en una aplicación móvil que, mediante conexión inalámbrica, se conecte a una placa colocada sobre el robot, conectada a éste mediante conexión USB, y mediante ordenes sea capaz de realizar diferentes acciones en el robot.

Fig. 3.1 Pioneer 3-DX con Raspberry Pi

La placa del robot actúa como servidor, mientras que la aplicación móvil actúa de cliente. Por lo tanto, cada uno de ellos tiene diferentes funcionalidades que deben ser cubiertas para el correcto funcionamiento del sistema.

3.2.1. Funcionalidades de la aplicación móvil

La idea principal que se ha utilizado para la elaboración del sistema ha sido la de dar la posibilidad al usuario tanto de manejar al robot de forma manual, como de seleccionar diferentes acciones que el usuario quiera que el robot ejecute. Por lo tanto, las funcionalidades principales de la aplicación móvil son las siguientes:

- Conexión mediante comunicación inalámbrica con la placa controladora.
- Control manual del robot.
- Control por acciones del robot.

La primera de las funcionalidades se llevará a cabo mediante conexión Wifi.

En cuanto al control manual, éste debe permitir al usuario desplazar al robot en todas las direcciones que éste permita.

Por otro lado, mediante el control por acciones, el usuario debe ser capaz de ejecutar diferentes acciones, tales como movimientos continuos, con distancias concretas, variaciones de velocidad o movimiento por coordenadas.

Estas funcionalidades han sido agrupadas en la siguiente tabla, y se identifican con el código FUA-XX, donde FUA significa *Funcionalidad Aplicación*, y XX es un número identificador de dos dígitos.

Funcionalidades de la aplicación móvil	
FUA-01	Conexión con el servidor mediante conexión Wifi.
FUA-02	Control manual: Movimiento frontal.
FUA-03	Control manual: Movimiento trasero.
FUA-04	Control manual: Giro a la derecha
FUA-05	Control manual: Giro a la izquierda
FUA-06	Control por acción: Movimiento frontal continuo.
FUA-07	Control por acción: Detención.
FUA-08	Control por acción: Giro continuo a la izquierda.
FUA-09	Control por acción: Giro continuo a la derecha.
FUA-10	Control por acción: Aceleración.
FUA-11	Control por acción: Freno.
FUA-12	Control por acción: Movimiento trasero continuo.
FUA-13	Control por acción: Movimiento frontal con distancia.
FUA-14	Control por acción: Movimiento trasero con distancia.
FUA-15	Control por acción: Movimiento continuo con ángulo.
FUA-16	Control por acción: Movimiento con distancia y ángulo.
FUA-17	Control por acción: Giro con grados.
FUA-18	Control por acción: Movimiento por coordenadas.

TABLA 3.1 FUNCIONALIDADES DE LA APLICACIÓN MÓVIL

3.2.2. Funcionalidades del servidor

En el lado del servidor, se encuentran todas las funcionalidades que se relacionan con la comunicación tanto con la aplicación como con el robot, así como las operaciones que conlleva cada acción para que esta se ejecute correctamente.

Estas funcionalidades se han identificado con el código FUS-XX, donde FUS significa *Funcionalidad Servidor*, y XX es un número identificador de dos dígitos, y están representadas en la siguiente tabla:

Funcionalidades del servidor	
FUS-01	Conexión con la aplicación móvil mediante Wifi.

Funcionalidades del servidor	
FUS-02	Conexión con el robot mediante el uso de la librería ARIA.
FUS-03	Recepción e interpretación de identificador de acción y parámetros de ejecución.

TABLA 3.2 FUNCIONALIDADES DEL SERVIDOR

3.3. Identificación de Requisitos de usuario

En este apartado se han agrupado todos los requisitos que debe reunir el sistema en base a las necesidades que ha considerado el usuario, y a las funcionalidades listadas en los apartados 3.2.1 y 3.2.2 del presente documento. Estos requisitos se han ido completando a lo largo del desarrollo del proyecto, añadiendo diferentes apartados que se han ido considerando importantes.

El catálogo de los requisitos de usuario ha sido desarrollado siguiendo el siguiente formato de tabla:

Identificador	
Nombre	
Descripción	
Fuente	
Necesidad	**Verificabilidad**
☐ Esencial ☐ Deseable ☐ Opcional	☐ Alta ☐ Media ☐ Baja

TABLA 3.3 MODELO DE TABLA DE REQUISITOS DE USUARIO

Los campos presentes en la tabla son los siguientes:

- **Identificador**: Cada requisito tendrá un identificador único. Este tendrá el formato “RU-XX”, donde “RU” hace referencia a “Requisito de Usuario” y “XX” será un número identificativo de dos dígitos.
- **Nombre**: Frase breve que define el contenido del requisito.
- **Descripción**: Definición completa del requisito.
- **Fuente**: Muestra el origen del requisito, pudiendo ser del tutor o del alumno del TFG.
- **Necesidad**: Determina el grado de importancia de aplicar un requisito.
    - Esencial: El requisito debe aplicarse siempre.

- Deseable: El requisito no es imprescindible, pero sería conveniente aplicarlo.
- Opcional: La aplicación del requisito no es necesaria para el funcionamiento principal del sistema. Sin embargo, podría ser útil aplicarlo.

- **Verificabilidad**: Determina el grado de posibilidad de comprobar que el requisito se ha aplicado en el software, y puede ser alta, media o baja.

3.3.1. Identificación de los Requisitos de usuario

RU-01	
Nombre	Conexión con el servidor
Descripción	El usuario deberá poder conectarse al servidor especificando su dirección IP y el puerto en el que se esté ejecutando el servidor.
Fuente	Alumno
Necesidad	**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.4 RU-01

RU-02	
Nombre	Elección de control manual o acciones
Descripción	El usuario deberá poder escoger entre utilizar el control manual o el control por acciones.
Fuente	Tutor
Necesidad	**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.5 RU-02

RU-03	
Nombre	Control manual: Movimiento frontal
Descripción	El usuario deberá poder desplazar mediante el control manual al robot frontalmente.

RU-03	
Fuente	Alumno
Necesidad	**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.6 RU-03

RU-04	
Nombre	Control manual: Movimiento trasero
Descripción	El usuario deberá poder desplazar mediante el control manual al robot marcha atrás.
Fuente	Alumno
Necesidad	**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.7 RU-04

RU-05	
Nombre	Control manual: Giro a la derecha
Descripción	El usuario deberá poder girar mediante el control manual al robot hacia la derecha.
Fuente	Alumno
Necesidad	**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.8 RU-05

RU-06	
Nombre	Control manual: Giro a la izquierda
Descripción	El usuario deberá poder girar mediante el control manual al robot hacia la izquierda.
Fuente	Alumno
Necesidad	**Verificabilidad**

RU-06	
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.9 RU-06

RU-07	
Nombre	Control por acción: Movimiento frontal continuo
Descripción	El usuario deberá poder desplazar mediante el control por acción al robot frontalmente de forma continua.
Fuente	Alumno
Necesidad	**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.10 RU-07

RU-08	
Nombre	Control por acción: Detención
Descripción	El usuario deberá poder detener mediante el control por acción al robot en cualquier momento, independientemente del movimiento que esté ejecutando.
Fuente	Alumno
Necesidad	**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.11 RU-08

RU-09	
Nombre	Control por acción: Giro continuo a la izquierda
Descripción	El usuario deberá poder girar mediante el control por acción al robot hacia la izquierda de forma continua.
Fuente	Alumno
Necesidad	**Verificabilidad**

RU-09	
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.12 RU-09

RU-10	
Nombre	Control por acción: Giro continuo a la derecha
Descripción	El usuario deberá poder girar mediante el control por acción al robot hacia la derecha de forma continua.
Fuente	Alumno
Necesidad	**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.13 RU-10

RU-11	
Nombre	Control por acción: Aceleración
Descripción	El usuario deberá poder aumentar la velocidad del robot mediante el control por acción.
Fuente	Tutor
Necesidad	**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.14 RU-11

RU-12	
Nombre	Control por acción: Freno
Descripción	El usuario deberá poder disminuir la velocidad del robot mediante el control por acción.
Fuente	Tutor
Necesidad	**Verificabilidad**

RU-12	
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.15 RU-12

RU-13	
Nombre	Control por acción: Movimiento trasero continuo
Descripción	El usuario deberá poder desplazar mediante el control por acción al robot marcha atrás de forma continua.
Fuente	Alumno
Necesidad	**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.16 RU-13

RU-14	
Nombre	Control por acción: Movimiento frontal con distancia
Descripción	El usuario deberá poder desplazar mediante el control por acción al robot frontalmente una distancia fija escogida por el propio usuario.
Fuente	Alumno
Necesidad	**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.17 RU-14

RU-15	
Nombre	Control por acción: Movimiento trasero con distancia
Descripción	El usuario deberá poder desplazar mediante el control por acción al robot marcha atrás una distancia fija escogida por el propio usuario.
Fuente	Alumno
Necesidad	**Verificabilidad**

RU-15	
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.18 RU-15

RU-16	
Nombre	Control por acción: Movimiento continuo con ángulo
Descripción	El usuario deberá poder desplazar mediante el control por acción al robot frontalmente de forma continua, y con un ángulo fijo escogido por el propio usuario
Fuente	Alumno
Necesidad	**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.19 RU-16

RU-17	
Nombre	Control por acción: Movimiento con distancia y ángulo
Descripción	El usuario deberá poder desplazar mediante el control por acción al robot frontalmente con un ángulo, con una distancia y ángulo fijos escogidos por el propio usuario
Fuente	Alumno
Necesidad	**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.20 RU-17

RU-18	
Nombre	Control por acción: Giro con grados
Descripción	El usuario deberá poder girar mediante el control por acción al robot unos grados fijos marcados por el usuario.
Fuente	Alumno
Necesidad	**Verificabilidad**

RU-18	
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.21 RU-18

RU-19	
Nombre	Control por acción: Movimiento por coordenadas
Descripción	El usuario deberá poder enviarle al robot unas coordenadas para que éste se desplace hasta ellas.
Fuente	Tutor
Necesidad	**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.22 RU-19

3.4. Casos de uso y requisitos del sistema

Tras la identificación de las funcionalidades del sistema en el apartado 3.2 del presente documento, se han desarrollado los casos de uso, los cuales son útiles para mostrar de forma más específica las funcionalidades que puede utilizar el usuario con el sistema. Seguidamente, también se han desarrollado los requisitos de software que satisfacen cada uno de los requisitos de usuario y casos de uso.

3.4.1. Identificación de los Casos de uso

Habiendo identificado las funcionalidades del sistema, es posible describir estos mediante casos de uso. Para ello, se ha elaborado la siguiente tabla descriptiva:

Identificador	
Nombre	
Descripción	
Requisitos relacionados	
Precondiciones	

Identificador	
Postcondiciones	

TABLA 3.23 MODELO DE TABLA DE CASOS DE USO

Los campos de la tabla son los siguientes:

- **Identificador**: Campo que identifica cada uno de los casos de uso. Sigue el formato CU-XX, donde CU significa "Caso de uso", y XX es un número identificativo de dos cifras.
- **Nombre**: Nombre del caso de uso.
- **Descripción**: Definición clara del caso de uso.
- **Requisitos relacionados**: Requisitos de software con los cuales se relaciona el caso de uso.
- **Precondiciones**: Condiciones previas que deben ocurrir para que el caso de uso se lleve a cabo.
- **Postcondiciones**: Resultados obtenidos tras llevar a cabo el caso de uso.

Los casos de uso son los siguientes:

CU-01	
Nombre	Conexión con el servidor
Descripción	El usuario se conecta con el servidor rellenando los campos de dirección IP y puerto en los que el servidor se esté ejecutando.
Requisitos relacionados	RSF-02, RSF-03, RSF-04, RSF-05
Precondiciones	• Iniciar el sistema.
Postcondiciones	• El usuario puede elegir el modo de control del robot. • El servidor recibe la petición de conexión.

TABLA 3.24 CU-01

CU-02	
Nombre	Uso de control manual
Descripción	El usuario elige el tipo de control manual.
Requisitos relacionados	RSF-06

CU-02	
Precondiciones	• Conexión con el servidor.
Postcondiciones	• El usuario puede elegir entre las opciones de control manual.

TABLA 3.25 CU-02

CU-03	
Nombre	Uso de control por acciones
Descripción	El usuario elige el tipo de control por acciones.
Requisitos relacionados	RSF-06
Precondiciones	• Conexión con el servidor.
Postcondiciones	• El usuario puede elegir entre las opciones del control por acciones.

TABLA 3.26 CU-03

CU-04	
Nombre	Movimiento frontal con control manual.
Descripción	El usuario pulsa el botón para llevar a cabo un movimiento frontal. Si se suelta el botón, el robot deja de moverse.
Requisitos relacionados	RSF-07
Precondiciones	• Uso del control manual.
Postcondiciones	• El robot ejecuta un movimiento frontal continuo sin detenerse hasta que el usuario suelta el botón. En caso de choque, el robot seguirá intentando desplazarse hasta que el usuario suelte el botón.

TABLA 3.27 CU-04

CU-05	
Nombre	Movimiento marcha atrás con control manual.
Descripción	El usuario pulsa el botón para llevar a cabo un movimiento marcha atrás. Si se suelta el botón, el robot deja de moverse.

CU-05	
Requisitos relacionados	RSF-07
Precondiciones	• Uso del control manual.
Postcondiciones	• El robot ejecuta un movimiento marcha atrás continuo sin detenerse hasta que el usuario suelta el botón. En caso de choque, el robot seguirá intentando desplazarse hasta que el usuario suelte el botón.

TABLA 3.28 CU-05

CU-06	
Nombre	Giro a la izquierda con control manual.
Descripción	El usuario pulsa el botón para llevar a cabo un giro a la izquierda. Si se suelta el botón, el robot deja de girar.
Requisitos relacionados	RSF-07
Precondiciones	• Uso del control manual.
Postcondiciones	• El robot ejecuta un giro continuo a la izquierda sin detenerse hasta que el usuario suelta el botón.

TABLA 3.29 CU-06

CU-07	
Nombre	Giro a la derecha con control manual.
Descripción	El usuario pulsa el botón para llevar a cabo un giro a la derecha. Si se suelta el botón, el robot deja de girar.
Requisitos relacionados	RSF-07
Precondiciones	• Uso del control manual.
Postcondiciones	• El robot ejecuta un giro continuo a la derecha sin detenerse hasta que el usuario suelta el botón.

TABLA 3.30 CU-07

CU-08	
Nombre	Combinación de movimiento lineal y giro con control manual.

CU-08	
Descripción	El usuario pulsa uno de los botones de movimiento frontal o trasero a la vez que pulsa uno de los botones de giro.
Requisitos relacionados	RSF-07
Precondiciones	• Uso del control manual.
Postcondiciones	• El robot ejecuta un movimiento lineal junto con un giro continuos sin detenerse hasta que el usuario suelta el botón.

TABLA 3.31 CU-08

CU-09	
Nombre	Movimiento frontal continuo con control por acciones.
Descripción	El usuario pulsa el botón de la opción de movimiento frontal continuo.
Requisitos relacionados	RSF-08
Precondiciones	• Uso de control por acciones
Postcondiciones	• El robot ejecuta un movimiento frontal continuo sin detenerse.

TABLA 3.32 CU-09

CU-10	
Nombre	Detención de movimiento con control por acciones.
Descripción	El usuario pulsa el botón de la opción de detener el movimiento del robot.
Requisitos relacionados	RSF-08
Precondiciones	• Uso de control por acciones
Postcondiciones	• El robot detiene cualquiera de sus movimientos.

TABLA 3.33 CU-10

CU-11	
Nombre	Giro continuo a la izquierda con control por acciones.

CU-11	
Descripción	El usuario pulsa el botón de la opción de girar continuamente al robot a la izquierda.
Requisitos relacionados	RSF-08
Precondiciones	• Uso de control por acciones
Postcondiciones	• El robot ejecuta un giro a la derecha continuo sin detenerse.

TABLA 3.34 CU-11

CU-12	
Nombre	Giro continuo a la derecha con control por acciones.
Descripción	El usuario pulsa el botón de la opción de girar continuamente al robot a la derecha.
Requisitos relacionados	RSF-08
Precondiciones	• Uso de control por acciones
Postcondiciones	• El robot ejecuta un giro a la derecha continuo sin detenerse.

TABLA 3.35 CU-12

CU-13	
Nombre	Aceleración con control por acciones.
Descripción	El usuario pulsa el botón de la opción de aumentar la velocidad del robot.
Requisitos relacionados	RSF-08
Precondiciones	• Uso de control por acciones
Postcondiciones	• El robot aumenta su velocidad de movimiento.

TABLA 3.36 CU-13

CU-14	
Nombre	Freno con control por acciones.

CU-14	
Descripción	El usuario pulsa el botón de la opción de disminuir la velocidad del robot.
Requisitos relacionados	RSF-08
Precondiciones	• Uso de control por acciones
Postcondiciones	• El robot disminuye su velocidad de movimiento.

TABLA 3.37 CU-14

CU-15	
Nombre	Movimiento trasero continuo con control por acciones.
Descripción	El usuario pulsa el botón de la opción de movimiento trasero continuo.
Requisitos relacionados	RSF-08
Precondiciones	• Uso de control por acciones
Postcondiciones	• El robot ejecuta un movimiento marcha atrás continuo.

TABLA 3.38 CU-15

CU-16	
Nombre	Movimiento frontal con distancia con control por acciones.
Descripción	El usuario pulsa el botón de la opción de movimiento frontal con distancia, inserta un valor numérico y pulsa el botón de enviar.
Requisitos relacionados	RSF-08, RSF-09
Precondiciones	• Uso de control por acciones • Inserción de valor numérico
Postcondiciones	• El robot se desplaza frontalmente la distancia establecida por el usuario.

TABLA 3.39 CU-16

CU-17	
Nombre	Movimiento trasero con distancia con control por acciones.

CU-17	
Descripción	El usuario pulsa el botón de la opción de movimiento trasero con distancia, inserta un valor numérico y pulsa el botón de enviar.
Requisitos relacionados	RSF-08, RSF-10
Precondiciones	• Uso de control por acciones • Inserción de valor numérico
Postcondiciones	• El robot se desplaza marcha atrás la distancia establecida por el usuario.

TABLA 3.40 CU-17

CU-18	
Nombre	Movimiento continuo con ángulo con control por acciones.
Descripción	El usuario pulsa el botón de la opción de movimiento continuo con ángulo, inserta un valor numérico y pulsa el botón de enviar.
Requisitos relacionados	RSF-08, RSF-11
Precondiciones	• Uso de control por acciones • Inserción de valor numérico
Postcondiciones	• El robot mantiene un movimiento frontal constante, mientras realiza el giro angular, y posteriormente mantiene su movimiento frontal.

TABLA 3.41 CU-18

CU-19	
Nombre	Movimiento con distancia y ángulo con control por acciones.
Descripción	El usuario pulsa el botón de la opción de movimiento con distancia y ángulo, inserta valores numéricos para la distancia y ángulo, y pulsa el botón de enviar.
Requisitos relacionados	RSF-08, RSF-12
Precondiciones	• Uso de control por acciones • Inserción de valores numéricos
Postcondiciones	• El robot se desplaza frontalmente la distancia fijada por el usuario, a la vez que realiza el giro.

TABLA 3.42 CU-19

CU-20	
Nombre	Giro con grados con control por acciones.
Descripción	El usuario pulsa el botón de la opción de giro con grados, inserta un valor numérico y pulsa el botón de enviar.
Requisitos relacionados	RSF-08, RSF-13
Precondiciones	• Uso de control por acciones • Inserción de valor numérico
Postcondiciones	• El robot gira los grados que el usuario ha marcado.

TABLA 3.43 CU-20

CU-21	
Nombre	Movimiento por coordenadas con control por acciones.
Descripción	El usuario pulsa el botón de la opción de movimiento por coordenadas, inserta dos valores numéricos representativos de las coordenadas, y pulsa el botón de enviar.
Requisitos relacionados	RSF-08, RSF-14
Precondiciones	• Uso de control por acciones • Inserción de valores numéricos
Postcondiciones	• El robot se desplaza hasta las coordenadas marcadas por el usuario, primero realizando un giro hasta colocarse de frente a las coordenadas, y posteriormente desplazándose frontalmente.

TABLA 3.44 CU-21

3.4.2. Identificación de los Requisitos del sistema

En este apartado se desarrollan los requisitos de software que satisfacen cada uno de los requisitos de usuario y los casos de uso. Se ha seguido el siguiente formato de tabla:

Identificador	
Nombre	
Descripción	
Fuente	
Necesidad	**Verificabilidad**
☐ Esencial ☐ Deseable ☐ Opcional	☐ Alta ☐ Media ☐ Baja

TABLA 3.45 MODELO DE TABLA DE REQUISITOS DE SISTEMA

Los campos presentes en la tabla son los siguientes:

- **Identificador**: Campo que identifica cada uno de los requisitos del sistema. Sigue el formato RSY-XX, donde RS significa "Requisito del Sistema", Y puede tomar los valores "F" o "NF", significando estos "Funcional" y "No funcional" respectivamente, y XX es un número identificador.
- **Nombre**: Frase breve que define el contenido del requisito.
- **Descripción**: Definición completa del requisito.
- **Fuente**: Muestra la procedencia del requisito, pudiendo ser del tutor o del alumno del TFG.
- **Necesidad**: Determina el grado de importancia de aplicar un requisito.
    - Esencial: El requisito debe aplicarse siempre.
    - Deseable: El requisito no es imprescindible, pero sería conveniente aplicarlo.
    - Opcional: La aplicación del requisito no es necesaria para el funcionamiento principal del sistema. Sin embargo, podría ser útil aplicarlo.
- **Verificabilidad**: Determina el grado de posibilidad de comprobar que el requisito se ha aplicado en el software, y puede ser alta, media o baja.

3.4.2.1. Requisitos funcionales

RSF-01	
Nombre	Actividad inicial
Descripción	El sistema dispondrá de una actividad inicial en la que se muestre el logo del sistema y, tras 3 segundos, pase a la actividad de conexión con el servidor.
Fuente	Alumno

Necesidad	Verificabilidad
☐ Esencial ☐ Deseable ☑ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.46 RSF-01

RSF-02	
Nombre	Actividad de conexión con el servidor
Descripción	El sistema dispondrá de una actividad de conexión, mediante la cual el usuario será capaz de conectarse con el servidor. El usuario deberá rellenar los siguientes campos de manera obligatoria: • Dirección IP • Puerto
Fuente	Alumno

Necesidad	Verificabilidad
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.47 RSF-02

RSF-03	
Nombre	Campo Dirección IP
Descripción	El campo Dirección IP será un String de 15 caracteres como máximo y 7 como mínimo. Se compondrá de 4 números con valores entre 0 y 255, separados por un total de 3 puntos.
Fuente	Alumno

Necesidad	Verificabilidad

RSF-03	
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.48 RSF-03

RSF-04	
Nombre	Campo Puerto
Descripción	El campo Puerto será un entero con un valor entre 0 y 65535.
Fuente	Alumno
Necesidad	**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.49 RSF-04

RSF-05	
Nombre	Realizar conexión con el servidor.
Descripción	La aplicación realizará la conexión con el servidor mediante la dirección IP y el Puerto establecidos.
Fuente	Alumno
Necesidad	**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.50 RSF-05

RSF-06	
Nombre	Actividad de selección de modo de control
Descripción	El usuario podrá seleccionar el modo mediante el cual quiera manejar el robot: Control manual o Control por acciones.
Fuente	Alumno
Necesidad	**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.51 RSF-06

RSF-07	
Nombre	Actividad de Control manual
Descripción	El sistema deberá permitir al usuario desplazar de forma lineal al robot, tanto frontal como marcha atrás, y girar en ambos sentidos mientras que los botones estén presionados. Al soltarlos, el robot deberá dejar de producir el movimiento correspondiente. Además, el sistema deberá permitir pulsar botones de forma simultánea.
Fuente	Alumno

Necesidad	Verificabilidad
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.52 RSF-07

RSF-08	
Nombre	Actividad de Control por acciones
Descripción	El sistema permitirá al usuario ejecutar determinadas acciones sobre el robot. Las acciones disponibles son las siguientes: • Movimiento frontal continuo • Detención • Giro continuo a la izquierda • Giro continuo a la derecha • Aceleración • Freno • Movimiento trasero continuo • Movimiento frontal con distancia • Movimiento trasero con distancia • Movimiento continuo con ángulo • Movimiento con distancia y ángulo • Giro con grados • Movimiento por coordenadas Cada una de estas acciones dispondrá de un botón propio para ser ejecutadas sobre el robot. Las acciones dependientes de un

RSF-08	
	valor para ejecutarse dispondrán de actividades propias.
Fuente	Alumno y tutor

Necesidad	Verificabilidad
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.53 RSF-08

RSF-09	
Nombre	Actividad de Movimiento frontal con distancia
Descripción	El sistema permitirá al usuario decidir el valor de la distancia que deberá recorrer el robot frontalmente. El valor de la distancia se medirá en milímetros.
Fuente	Alumno y tutor

Necesidad	Verificabilidad
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.54 RSF-09

RSF-10	
Nombre	Actividad de Movimiento trasero con distancia
Descripción	El sistema permitirá al usuario decidir el valor de la distancia que deberá recorrer el robot marcha atrás. El valor de la distancia se medirá en milímetros.
Fuente	Alumno y tutor

Necesidad	Verificabilidad
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.55 RSF-10

RSF-11	
Nombre	Actividad de Movimiento continuo con ángulo
Descripción	El sistema permitirá al usuario decidir el valor del ángulo que

RSF-11	
	deberá girar el robot durante el movimiento continuo. El valor del ángulo se medirá en grados, pudiendo ser un valor negativo.
Fuente	Alumno y tutor
Necesidad	**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.56 RSF-11

RSF-12	
Nombre	Actividad de Movimiento con distancia y ángulo
Descripción	El sistema permitirá al usuario decidir los valores de la distancia y el ángulo que deberá recorrer el robot. El valor de la distancia se medirá en milímetros, y el valor del ángulo se medirá en grados, pudiendo ser un valor negativo.
Fuente	Alumno y tutor
Necesidad	**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.57 RSF-12

RSF-13	
Nombre	Actividad de Giro con grados
Descripción	El sistema permitirá al usuario decidir el valor del ángulo que deberá girar el robot. El valor del ángulo se medirá en grados, pudiendo ser un valor negativo.
Fuente	Alumno y tutor
Necesidad	**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.58 RSF-13

RSF-14		
Nombre	Actividad de Movimiento por coordenadas	
Descripción	El sistema permitirá al usuario decidir los valores de las coordenadas a las que debe trasladarse el robot. Los valores estarán medidos en milímetros, pudiendo ser ambos valores negativos.	
Fuente	Tutor	
Necesidad		**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional		☑ Alta ☐ Media ☐ Baja

TABLA 3.59 RSF-14

RSF-15		
Nombre	Envío y recepción de mensajes en el sistema	
Descripción	La aplicación deberá poder enviar mensajes diferenciados al servidor dependiendo de la acción a ejecutar.	
Fuente	Alumno	
Necesidad		**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional		☑ Alta ☐ Media ☐ Baja

TABLA 3.60 RSF-15

RSF-16		
Nombre	Detención de movimiento	
Descripción	El sistema deberá permitir al usuario detener el movimiento del robot en cualquier momento, independientemente de la acción que el robot esté ejecutando.	
Fuente	Alumno	
Necesidad		**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional		☑ Alta ☐ Media ☐ Baja

TABLA 3.61 RSF-16

3.4.2.2.Requisitos No Funcionales

RSNF-01	
Nombre	Tiempo de respuesta del sistema
Descripción	El sistema debe tener un tiempo de respuesta inferior a 1,5 segundos.
Fuente	Alumno
Necesidad	**Verificabilidad**
☐ Esencial ☐ Deseable ☑ Opcional	☐ Alta ☑ Media ☐ Baja

TABLA 3.62 RSNF-01

RSNF-02	
Nombre	Compatibilidad de dispositivos
Descripción	El sistema deberá funcionar correctamente en sistemas Android con versión 6.0 *Marshmallow*.
Fuente	Alumno
Necesidad	**Verificabilidad**
☐ Esencial ☑ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.63 RSNF-02

RSNF-03	
Nombre	Uso del botón de retroceso
Descripción	Utilizar el botón de retroceso de Android debe permitir cerrar la actividad que esté siendo visualizada en ese momento, volviendo a la actividad anterior.
Fuente	Alumno
Necesidad	**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.64 RSNF-03

RSNF-04	
Nombre	Integridad de los mensajes de sistema
Descripción	Será necesaria la utilización de protocolos para asegurar que los mensajes de entrada y salida del sistema llegarán íntegramente tanto al servidor como al terminal. Para esto, se utilizará el protocolo TCP de conexión a internet.
Fuente	Alumno

Necesidad	Verificabilidad
☑ Esencial ☐ Deseable ☐ Opcional	☐ Alta ☑ Media ☐ Baja

TABLA 3.65 RSNF-04

RSNF-05	
Nombre	Mensaje de error tras fallo al rellenar formularios
Descripción	El sistema deberá mostrar un mensaje de error al usuario en caso de que éste rellene algún formulario de forma incorrecta.
Fuente	Alumno

Necesidad	Verificabilidad
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.66 RSNF-05

RSNF-06	
Nombre	Mensaje de error por fallo de conexión con el servidor
Descripción	El sistema deberá mostrar un mensaje de error al usuario en caso de que exista algún error en la conexión del sistema con el servidor.
Fuente	Alumno

Necesidad	Verificabilidad
☑ Esencial ☐ Deseable ☐ Opcional	☑ Alta ☐ Media ☐ Baja

TABLA 3.67 RSNF-06

RSNF-07		
Nombre	Desconexión de la aplicación	
Descripción	El sistema deberá permitir la desconexión de la aplicación, permitiendo al cliente reconectarse posteriormente.	
Fuente	Alumno	
Necesidad		**Verificabilidad**
☑ Esencial ☐ Deseable ☐ Opcional		☑ Alta ☐ Media ☐ Baja

TABLA 3.68 RSNF-07

3.5.Análisis y validación de requisitos

En este apartado se presenta la trazabilidad existente entre los requisitos de sistema funcionales y los requisitos de usuario, y la trazabilidad entre los casos de uso y los requisitos de sistema. Estas tablas muestran como para cada requisito de usuario y cada caso de uso, existe al menos un requisito funcional del sistema que cubre su funcionamiento.

3.5.1. Trazabilidad de requisitos de usuario y de sistema

Sistema / Usuario	RSF-01	RSF-02	RSF-03	RSF-04	RSF-05	RSF-06	RSF-07	RSF-08	RSF-09	RSF-10	RSF-11	RSF-12	RSF-13	RSF-14	RSF-15	RSF-16
RU-01		X	X	X	X										X	
RU-02						X										
RU-03							X								X	X
RU-04							X								X	X
RU-05							X								X	X
RU-06							X								X	X
RU-07								X							X	X
RU-08								X							X	X
RU-09								X							X	X
RU-10								X							X	X
RU-11								X							X	X
RU-12								X							X	X
RU-13								X							X	X
RU-14								X	X						X	X
RU-15								X		X					X	X
RU-16								X			X				X	X
RU-17								X				X			X	X
RU-18								X					X		X	X
RU-19								X						X	X	X

TABLA 3.69 MATRIZ DE TRAZABILIDAD DE REQUISITOS DE USUARIO Y SISTEMA

3.5.2. Trazabilidad de casos de uso y requisitos de sistema

Sistema / Usuario	RSF-01	RSF-02	RSF-03	RSF-04	RSF-05	RSF-06	RSF-07	RSF-08	RSF-09	RSF-10	RSF-11	RSF-12	RSF-13	RSF-14	RSF-15	RSF-16
CU-01		X	X	X	X										X	
CU-02						X										
CU-03						X										
CU-04							X								X	X
CU-05							X								X	X
CU-06							X								X	X
CU-07							X								X	X
CU-08							X								X	X
CU-09								X							X	X
CU-10								X							X	X
CU-11								X							X	X
CU-12								X							X	X
CU-13								X							X	X
CU-14								X							X	X
CU-15								X							X	X
CU-16								X	X						X	X
CU-17								X		X					X	X
CU-18								X			X				X	X
CU-19								X				X			X	X
CU-20								X					X		X	X
CU-21								X						X	X	X

TABLA 3.70 MATRIZ DE TRAZABILIDAD DE CASOS DE USO Y REQUISITOS DE SISTEMA

4. DISEÑO DEL SISTEMA

En el presente apartado se describe el diseño del sistema con el objetivo de mostrar, de forma clara, como se ha desarrollado el sistema para resolver todo el problema planteado en el apartado 3 del presente documento.

Para ello, inicialmente se ha elaborado un apartado en el que describir la arquitectura del sistema. Posteriormente, se aporta una revisión de la interfaz de usuario, mediante la cual se busca mostrar de forma más clara el diseño de la aplicación móvil, así como su funcionamiento. Por último, se ha redactado un apartado de implementación, en el que se muestran diferentes apartados acerca de cómo se ha desarrollado el sistema.

4.1.Arquitectura del sistema

El sistema de información diseñado tiene un cometido principal, que se basa en el envío de datos al servidor para que este sea capaz de identificar la orden del usuario sobre la interfaz. Debido a esto, el sistema se ha dividido en varios niveles, de manera que el desarrollo pueda dividirse en partes más pequeñas. Los componentes físicos son los siguientes:

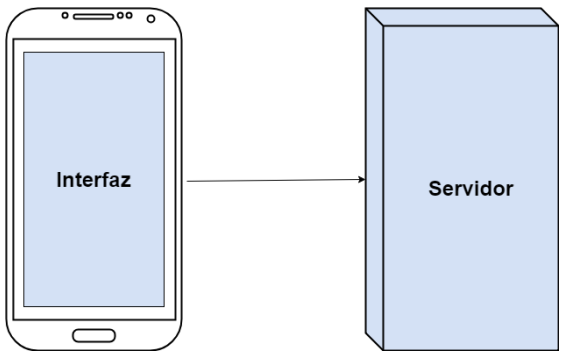

Fig. 4.1 Diagrama de componentes físicos

Como puede apreciarse en la imagen, el sistema solo consta de un terminal móvil ejecutando la aplicación, y de un servidor, el cual se situará en la placa colocada sobre el robot. El servidor no requiere de una base de datos, ya que no es necesario almacenar nada.

4.1.1. Definición de niveles de arquitectura

En este apartado se distinguen los niveles de arquitectura que componen el sistema. Se han establecido 3 niveles distintos de arquitectura:

- **Nivel de interfaz**: En el nivel de interfaz se encuentra todo lo relacionado con el diseño *front-end* del sistema. El usuario accede al sistema mediante este nivel, navegando por las diferentes actividades de la aplicación mediante los botones y campos establecidos en la interfaz.
- **Nivel de control**: En el nivel de control se puede encontrar todo lo relacionado con la gestión de las peticiones del nivel de interfaz para realizar una comunicación con el servidor.
- **Nivel de servidor**: En este último nivel se encuentra el servidor como tal. Es el encargado de gestionar las peticiones del nivel de control y ejecutar la acción correspondiente sobre el robot. Dependiendo de la petición, el servidor puede necesitar información adicional aportada por el usuario.

4.1.2. Especificación del entorno tecnológico

En este apartado se muestran las herramientas utilizadas para el desarrollo del proyecto de forma resumida, en base a lo desarrollado en el apartado 2 del presente documento.

- **Herramientas de desarrollo**

Tipo	Nombre	Versión
Herramienta de desarrollo de la aplicación móvil	Android Studio	3.1.3
Lenguaje de desarrollo de la aplicación móvil	Java	1.8.0_151
Herramienta de desarrollo del servidor	Eclipse – Notepad++ Ubuntu Mate - ARIA	Oxygen 4.7.0 - 6.9.1 16.04 – 2.9.1
Lenguaje de desarrollo del servidor	Java	1.8.0_151

TABLA 4.1 HERRAMIENTAS DE DESARROLLO

- **Componentes físicos**

En este apartado se agrupan todos los componentes físicos utilizados en el desarrollo del sistema.

Robot	
Nombre	Pioneer 3-DX

Robot	
Peso	9 kg
Velocidad máxima frontal	1.2 m/s
Radio de giro	26.7 cm
Máxima pendiente	25%
Tiempo de autonomía	8-10 horas
Terrenos transitables	• Azulejo • Baldosas • Pavimento

TABLA 4.2 COMPONENTE FÍSICO: ROBOT

Placa controladora	
Nombre	Raspberry Pi 3 Model B
SoC	Broadcom BCM2837
CPU	1.4GHz 64-bit quad-core ARMv8
GPU	Broadcom VideoCore IV
Memoria RAM	1 GB
Almacenamiento	Ranura MicroSD
Puertos E/S	• 4 puertos USB 2.0 • Conector MIPI CSI • Conector RCA • HDMI • Jack 3.5 mm
Periféricos de bajo nivel	• 17 x GPIO • 1 x bus HAT ID
Conectividad de red	• Conector RJ-45 • Wifi 802.11n/ac

Placa controladora	
	• Bluetooth 4.2 BLE
Sistemas operativos soportados	• Debian (Raspbian) • Fedora (Pidora) • Arch Linux (Arch Linux ARM) • Slackware Linux • SUSE 65 Linux Enterprise Server for ARM • RISC OS2 • Windows 10
Consumo energético	800 mA

TABLA 4.3 COMPONENTE FÍSICO: PLACA

4.2.Revisión de la interfaz de usuario

En este apartado se desarrolla el diseño detallado del comportamiento de la interfaz de usuario a partir de lo especificado en el apartado 3 del presente documento.

Fig. 4.2 Logo de la aplicación móvil

La aplicación móvil desarrollada ha sido denominada *ARIAController*, el cual ya busca comunicar al usuario que el controlador no se limita únicamente a un tipo de robot, sino que es compatible con cualquiera de los robots analizados en el apartado 2.1 y que sean compatibles con la librería ARIA.

El logo se ha elaborado a partir del diseño del robot Pioneer 3-DX, ya que ha sido el utilizado para el desarrollo del proyecto.

Inicialmente, cuando el usuario abre la aplicación, lo primero que se muestra es la actividad inicial, en la cual, durante 2 segundos, se puede visualizar el logo.

Tras el paso de los segundos de la actividad inicial, la aplicación transita a la actividad de conexión, en la cual el usuario debe rellenar los campos de dirección IP y Puerto en los cuales se está ejecutando el servidor.

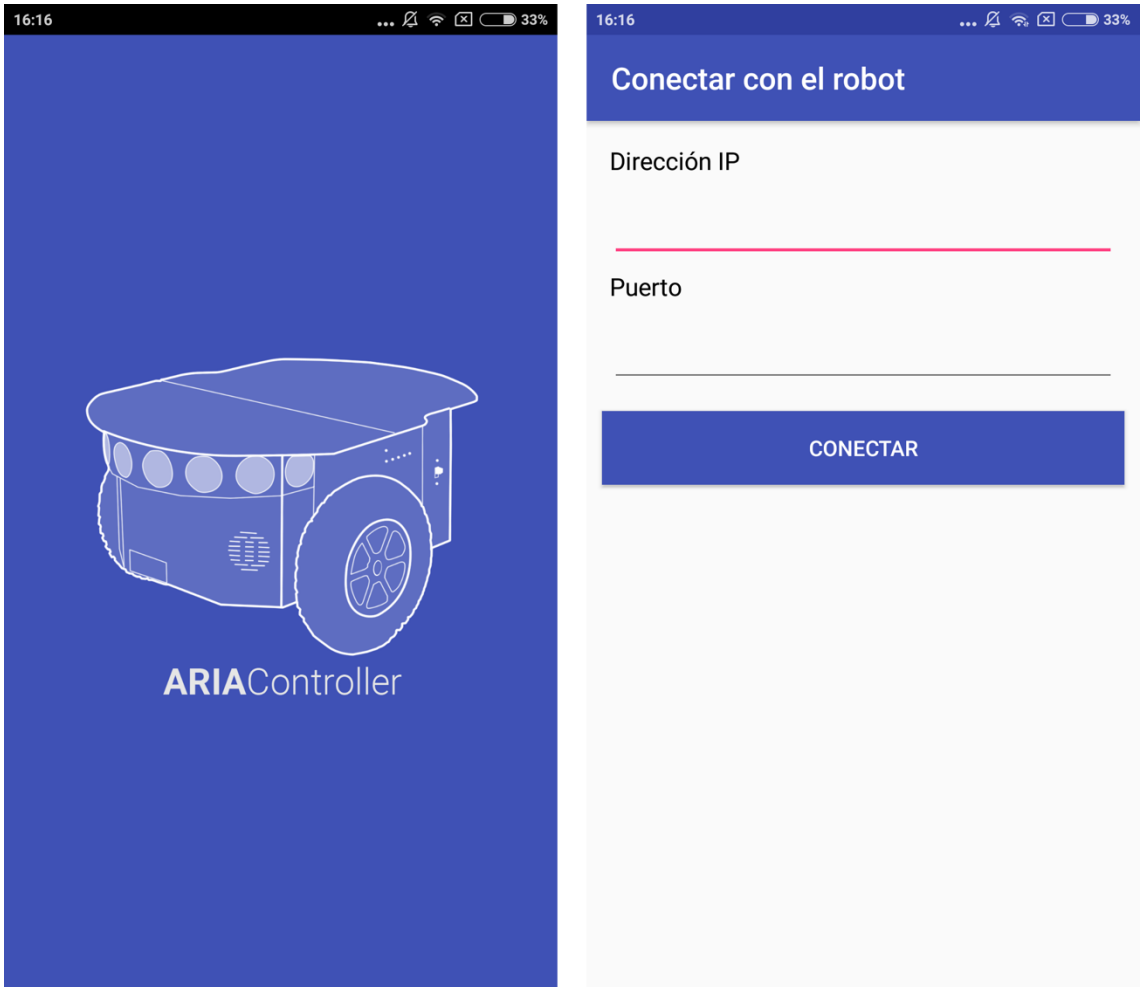

Fig. 4.3 Actividades inicial y de conexión

Tras rellenar los campos correctamente, el usuario puede realizar la conexión con el robot mediante la pulsación del botón *Conectar* y transitar a la actividad de selección de Modo de control. En esta actividad, el usuario dispone de dos botones claramente diferenciados, cada uno ocupando la mitad de la pantalla del terminal, y que se corresponden con los modos de Control manual y Control por acciones. Pulsar uno de ellos hace que la aplicación transite a su actividad correspondiente.

El primero de los botones es el de Control manual. Tras pulsarlo, se transita a su actividad correspondiente, en la cual se muestran 4 botones: Los dos primeros de

movimiento lineal frontal o marcha atrás, y los dos últimos correspondientes al giro del robot a cada uno de sus lados. Estos botones pueden ser pulsados a la vez y permiten al usuario realizar movimientos conjuntos, como avanzar mientras se gira para obtener un movimiento angular.

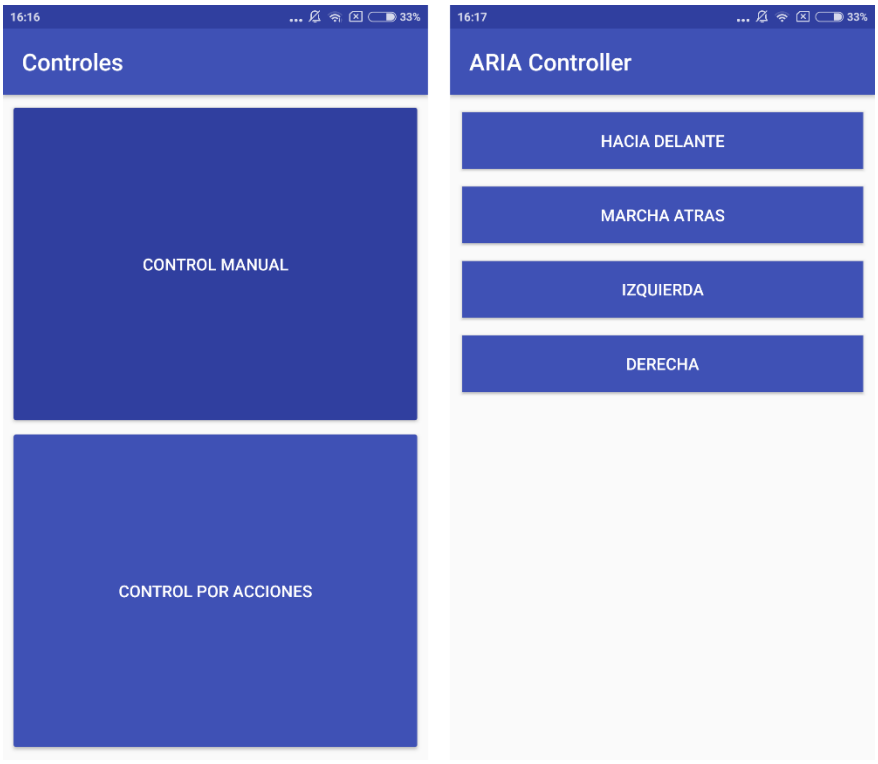

Fig. 4.4 Actividades de selección de modo de control y control manual

Por otro lado, desde la actividad de selección de modo de control, el usuario puede transitar también a la actividad de selección de acciones. En ella, se muestran las 13 diferentes acciones que pueden realizarse mediante este modo de control, y que están listadas en la TABLA 3.1 FUNCIONALIDADES DE LA APLICACIÓN MÓVIL del presente documento.

Mientras que algunas de estas acciones pueden ejecutarse directamente pulsando su botón correspondiente, también hay acciones que requieren de la introducción de valores por parte del usuario, como distancias o ángulos.

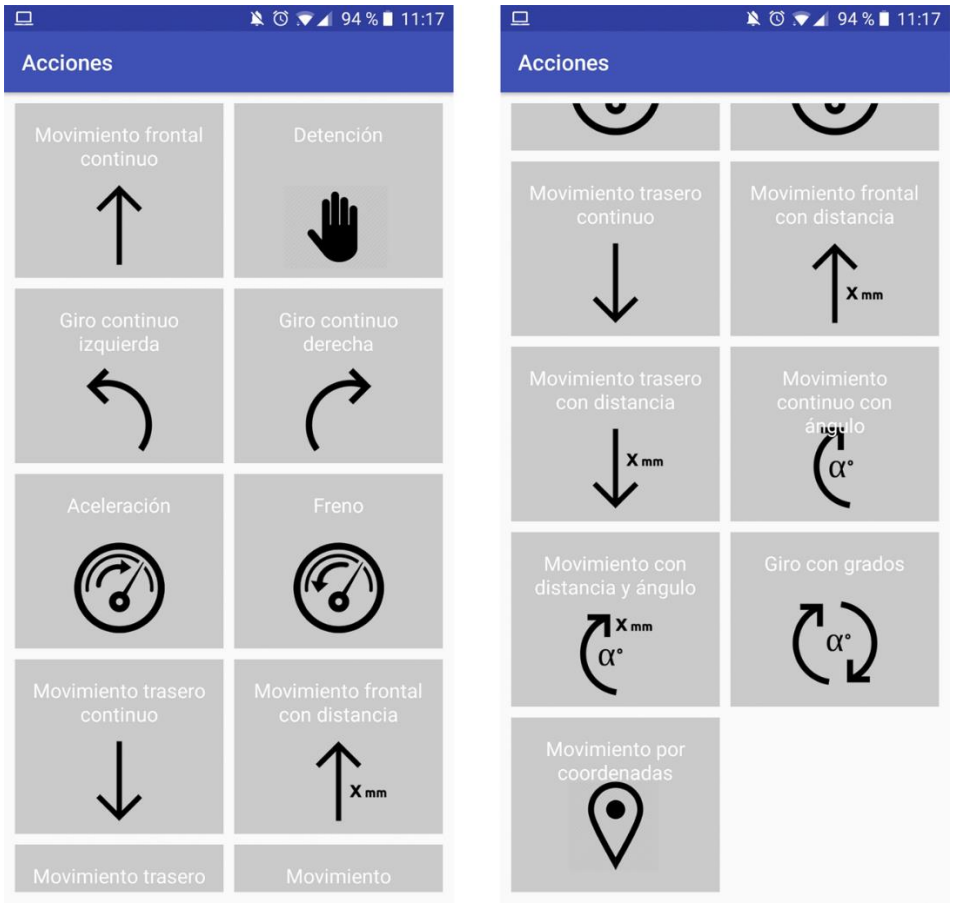

Fig. 4.5 Actividad de modo de Control por acciones

Las actividades de las acciones que requieren valores siguen el mismo formato que la actividad de conexión, con uno o más campos rellenables por el usuario y un botón mediante el cual enviar la información al servidor.

Dependiendo del campo, las unidades de medida de estos son distintas:

- **Distancias**: Medidas en milímetros.
- **Ángulos**: Medidas en grados sexagesimales.
- **Coordenadas**: Medidas en milímetros.

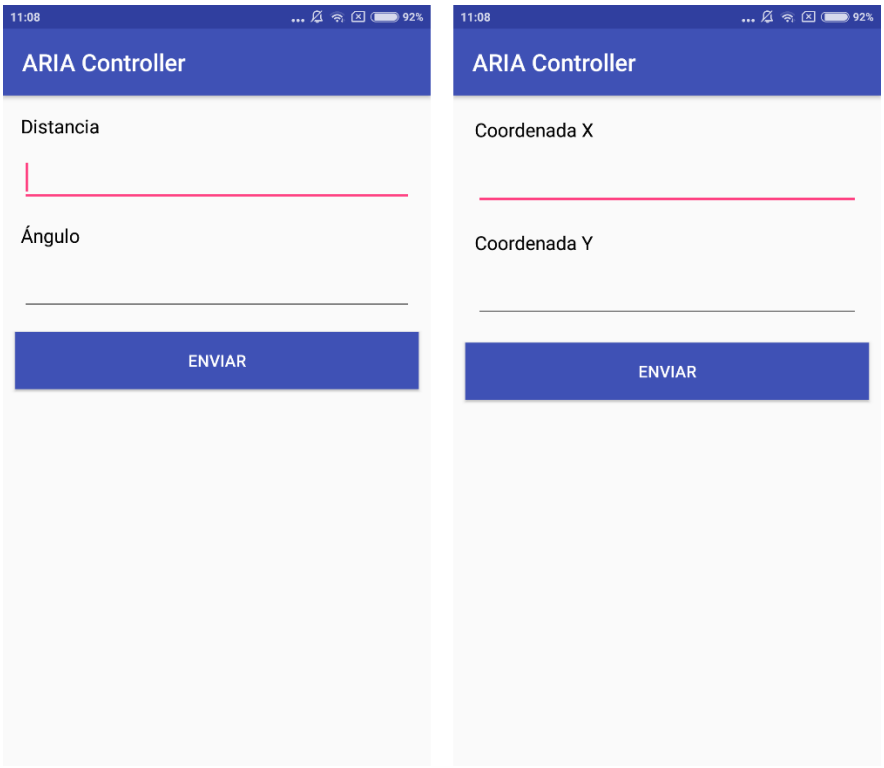

Fig. 4.6 Actividades de movimiento con distancia y ángulo y movimiento por coordenadas

4.3.Implementación del sistema

En este apartado se desarrolla el funcionamiento del sistema, con el objetivo de explicarlo de forma concisa.

Inicialmente, con el objetivo de conseguir la mayor compatibilidad posible, se decidió desarrollar el servidor en el lenguaje Java, igual que la aplicación móvil. Además, también se escogió como forma de conexión entre aplicación y servidor el uso de sockets, con el objetivo de simplificar lo máximo posible el funcionamiento del sistema.

Dado que estos sockets permiten el envío de diferentes tipos de mensajes, se escogió el tipo String para el envío de mensajes, ya que el protocolo TCP de la conexión a internet garantiza el envío completo del mensaje y su correcta recepción.

Por lo tanto, cada una de las acciones que el usuario puede realizar en la aplicación y que requiere una conexión con el servidor tiene un código individual que identifica la acción a realizar. Los códigos utilizados son los siguientes:

Código	Acción
1	Conexión inicial con el servidor

Código	Acción
2	Libre
3	Control manual 1: Movimiento frontal
4	Control manual 2: Movimiento trasero
5	Control manual 3: Giro a la izquierda
6	Control manual 4: Giro a la derecha
7	Libre
8	Libre
9	Libre
10	Control por acciones 1: Movimiento frontal continuo
11	Control por acciones 2: Movimiento frontal con distancia
12	Control por acciones 3: Movimiento trasero continuo
13	Control por acciones 4: Movimiento trasero con distancia
14	Control por acciones 5: Movimiento continuo con ángulo
15	Control por acciones 6: Movimiento con distancia y ángulo
16	Control por acciones 7: Giro continuo a la derecha
17	Control por acciones 8: Giro con grados
18	Control por acciones 9: Movimiento por coordenadas
19	Control por acciones 10: Detención
20	Control por acciones 11: Aceleración
21	Control por acciones 12: Freno
22	Control por acciones 13: Giro continuo a la izquierda

TABLA 4.4 CÓDIGOS DE ACCIÓN DEL SISTEMA

En esta tabla se pueden observar dos cosas: hay códigos libres y que el orden de las acciones no se corresponde con el orden mostrado anteriormente en la aplicación. La primera de estas observaciones se debe a que, inicialmente, se reservaron 10 números para cada modo de control, pero durante el avance del proyecto se observó que no eran

necesarios tantos códigos, dejándolos libres para una posible posterior actualización del sistema con más acciones.

Por otro lado, el desorden de las acciones en comparación con la aplicación se debe a que, al comienzo del desarrollo del sistema, se listaron las diferentes acciones sin seguir un orden especifico, el cual se mantuvo en el servidor. Sin embargo, en la aplicación fueron ordenadas para una mejor visualización de las acciones para el usuario, así como una mayor facilidad de control del robot.

Tras realizar diferentes pruebas con los sockets y evaluar las opciones de mantener el socket abierto por sesión, o bien abrirlo y cerrarlo por cada conexión, se decidió el uso de esta última forma, debido a que el usuario puede pasar tiempo sin enviar acciones al servidor, y de esta forma el servidor pasa a ser más eficiente. Sin embargo, utilizar de esta forma los sockets implica que la aplicación deba realizar conexiones con el servidor cada vez que el usuario decida realizar una acción. Por lo tanto, tras la conexión inicial, tanto la dirección IP como el Puerto son guardados en la interfaz *SharedPreferences* de la aplicación, con el objetivo de poder consultar esta información desde cualquier parte de la aplicación sin necesidad de estarla rellenando constantemente.

Para enviar la información necesaria al servidor para poder ejecutar las acciones, se han utilizado Strings con un formato concreto. Este formato se basa en juntar todos los datos necesarios unidos mediante el signo “;”, para que así el servidor pueda separar los valores sin problema. Por lo tanto, dependiendo de si la acción tiene o no la necesidad de más parámetros, el String puede estar compuesto de 1, 2 o 3 parámetros. Un ejemplo de este tipo de String es el siguiente:

15;2000;90;

En el ejemplo, se utiliza la acción 15, correspondiente al movimiento con distancia y ángulo, se envía una distancia de 2000 milímetros y un ángulo de 90º. Como ya se ha mencionado, el número de parámetros es dependiente del tipo de acción.

En cuanto al funcionamiento del servidor, la ejecución principal se basa en un bucle infinito, el cual comienza inicializando las dependencias de ARIA y creando el socket que se utilizará para cada conexión, quedándose escuchando por una conexión desde la aplicación. Tras recibirla, divide el String recibido y, en base al código de acción, ejecuta la correspondiente. Cada una de estas acciones requiere que las dependencias de

ARIA se hayan resuelto de forma correcta, ya que el robot es representado en Java mediante la clase ArRobot, presente en la librería de ARIA. Esta clase presenta todos los métodos necesarios para la ejecución de las acciones desarrolladas, a excepción del movimiento por coordenadas.

La ejecución de la tarea de movimiento por coordenadas ha requerido la elaboración de dos métodos diferentes para poder controlar el orden de ejecución de los movimientos del robot. El objetivo es simple: rotar al robot los grados necesarios para colocarlo frente a las coordenadas requeridas y, posteriormente, desplazar al robot frontalmente la distancia requerida.

5. PRUEBAS DEL SISTEMA

El objetivo principal del presente apartado es el de realizar un plan de pruebas acorde a lo requerido por el resto de los apartados del proyecto, asegurando un funcionamiento óptimo del sistema en los casos de uso que el usuario pueda ejecutar

Inicialmente se establece la especificación técnica del plan de pruebas, y posteriormente se aportan matrices de trazabilidad de los requisitos con respecto a las pruebas del sistema.

5.1.Especificación técnica del plan de pruebas

Para el desarrollo del plan de pruebas, se han utilizado los siguientes dispositivos móviles:

Xiaomi Redmi Note 4	
Características	**Descripción**
Sistema Operativo	Android 6.0 Marshmallow – MIUI 9.5
Memoria RAM	3 GB
Procesador	Mediatek Helio X20 Deca-core 2.11 GHz
N.º Procesadores	10
Resolución de pantalla	1080 x 1920 píxeles
Resolución de cámara	13 megapíxeles

TABLA 5.1 TERMINAL MÓVIL 1

Oneplus 5T	
Características	**Descripción**
Sistema Operativo	Android 8.1.0 Oreo – OxygenOS 5.1.5
Memoria RAM	8 GB
Procesador	Snapdragon 835
N.º Procesadores	8

Oneplus 5T	
Resolución de pantalla	1080 x 2160 píxeles
Resolución de cámara	16 megapíxeles

TABLA 5.2 TERMINAL MÓVIL 2

Se han escogido dos terminales con versión de Android igual o superior a 6.0 *Marshmallow*, con el objetivo de corroborar el funcionamiento del sistema en diferentes versiones del sistema operativo. Hay que aclarar que, pese a que el dispositivo Oneplus 5T posee una versión de Android *Oreo*, previamente se utilizó también con versiones de Android *Nougat* para realizar pruebas con el sistema, funcionando igual que con la versión actual.

Por otro lado, la placa utilizada ha sido la Raspberry Pi 3, y el robot el Pioneer 3-DX, cuyas especificaciones están redactadas en el apartado 4.1.2 del presente documento.

Se han definido los siguientes tipos de pruebas:

Tipos de prueba	
Prueba	**Descripción**
Pruebas del sistema	Van dirigidas a validar que el sistema cumple los requisitos de funcionamiento esperado, recogidos en el catálogo de requisitos de usuario, casos de uso y requisitos del sistema, y conseguir la aceptación final del sistema por parte del usuario.
Pruebas de implantación	Verifican que el sistema puede ser instalado en la plataforma requerida, y que el sistema funcionará correctamente cuando sea instalado. Además, comprueban que el sistema coexiste con el resto de los sistemas de la instalación.

TABLA 5.3 TIPOS DE PRUEBAS

5.2.Especificación técnica de niveles de prueba

5.2.1. Pruebas del sistema

El objetivo de las siguientes pruebas es validar que el sistema cumple con el funcionamiento esperado y permitir al cliente determinar su aceptación teniendo en cuenta la funcionalidad y el rendimiento. Para las pruebas se emplearán tablas con el siguiente formato:

Identificador	
Nombre	
Descripción	
Precondiciones	
Criterios de aceptación	
Requisitos relacionados	

TABLA 5.4 FORMATO DE TABLA DE PRUEBAS DE SISTEMA

Los campos presentes en la tabla son los siguientes:

- **Identificador**: Cada prueba tendrá un identificador único. Este tendrá el formato "PS-XX", donde "PS" hace referencia a "Prueba del Sistema" y "XX" será un número identificativo de dos dígitos.
- **Nombre**: Frase breve que define el contenido de la prueba.
- **Descripción**: Definición completa de la prueba.
- **Precondiciones**: Condiciones previas que deben ocurrir para llevar a cabo la prueba.
- **Criterios de aceptación**: Objetivos que debe cumplir la prueba para ser aceptada.
- **Requisitos relacionados**: Requisitos de usuario implicados en la prueba.

A continuación, se desarrollan las pruebas que verifican los requisitos de usuario, casos de uso y requisitos de sistema, situados en el apartado 3.3, 3.4.1 y 0 respectivamente en el presente documento. Para cada una de estas pruebas de sistema, se han elaborado

diferentes casos de ejecución para obtener todas las posibles soluciones para cada prueba. Estas posibilidades se han descrito en el apartado 5.2.3 del presente documento.

PS-01	
Nombre	Conexión con el servidor mediante conexión Wifi
Descripción	El usuario puede conectarse con el servidor mediante dirección IP y Puerto.
Precondiciones	1. Abrir la aplicación
Criterios de aceptación	La aplicación se conecta con el servidor y transita a la actividad de selección de modo.
Requisitos relacionados	RSF-02, RSF-03, RSF-04, RSF-05

TABLA 5.5 PS-01

PS-02	
Nombre	Uso de control manual
Descripción	El usuario puede acceder a los controles manuales desde la actividad de elección de modo de control.
Precondiciones	1. Abrir la aplicación 2. Conexión con el servidor
Criterios de aceptación	La aplicación transita correctamente a la actividad de control manual.
Requisitos relacionados	RSF-06, RSF-07

TABLA 5.6 PS-02

PS-03	
Nombre	Uso de control por acciones
Descripción	El usuario puede acceder a los controles por acciones desde la actividad de elección de modo de control.

PS-03	
Precondiciones	1. Abrir la aplicación 2. Conexión con el servidor
Criterios de aceptación	La aplicación transita correctamente a la actividad de control por acciones.
Requisitos relacionados	RSF-06, RSF-08

TABLA 5.7 PS-03

PS-04	
Nombre	Movimiento frontal mediante control manual
Descripción	El usuario puede ejecutar un movimiento frontal utilizando el control manual mientras mantenga pulsado el botón.
Precondiciones	1. Abrir la aplicación 2. Conexión con el servidor 3. Selección de modo de control manual
Criterios de aceptación	La aplicación manda correctamente el ID de la acción y el robot lleva a cabo la acción correctamente.
Requisitos relacionados	RSF-07

TABLA 5.8 PS-04

PS-05	
Nombre	Movimiento trasero mediante control manual
Descripción	El usuario puede ejecutar un movimiento trasero utilizando el control manual mientras mantenga pulsado el botón.
Precondiciones	1. Abrir la aplicación 2. Conexión con el servidor 3. Selección de modo de control manual
Criterios de aceptación	La aplicación manda correctamente el ID de la acción y el robot lleva a cabo la acción correctamente.

PS-05	
Requisitos relacionados	RSF-07

TABLA 5.9 PS-05

PS-06	
Nombre	Giro a la derecha mediante control manual
Descripción	El usuario puede ejecutar un giro a la derecha utilizando el control manual mientras mantenga pulsado el botón.
Precondiciones	1. Abrir la aplicación 2. Conexión con el servidor 3. Selección de modo de control manual
Criterios de aceptación	La aplicación manda correctamente el ID de la acción y el robot lleva a cabo la acción correctamente.
Requisitos relacionados	RSF-07

TABLA 5.10 PS-06

PS-07	
Nombre	Giro a la izquierda mediante control manual
Descripción	El usuario puede ejecutar un giro a la izquierda utilizando el control manual mientras mantenga pulsado el botón.
Precondiciones	1. Abrir la aplicación 2. Conexión con el servidor 3. Selección de modo de control manual
Criterios de aceptación	La aplicación manda correctamente el ID de la acción y el robot lleva a cabo la acción correctamente.
Requisitos relacionados	RSF-07

TABLA 5.11 PS-07

PS-08	
Nombre	Uso combinado de movimiento lineal y angular
Descripción	El usuario puede ejecutar un movimiento lineal, tanto frontal como trasero, y un giro hacia cualquiera de los lados del robot a la vez, con el objetivo de generar un movimiento circular en el robot.
Precondiciones	1. Abrir la aplicación 2. Conexión con el servidor 3. Selección de modo de control manual
Criterios de aceptación	La aplicación manda correctamente los IDS de las acciones, y el robot lleva a cabo ambas acciones en conjunto correctamente.
Requisitos relacionados	RSF-07

TABLA 5.12 PS-08

PS-09	
Nombre	Movimiento frontal continuo mediante control por acciones
Descripción	El usuario puede ejecutar un movimiento frontal continuo utilizando el control por acciones. Para ello, el usuario podrá pulsar el botón de la acción correspondiente.
Precondiciones	1. Abrir la aplicación 2. Conexión con el servidor 3. Selección de modo de control por acciones
Criterios de aceptación	La aplicación manda correctamente el ID de la acción y el robot lleva a cabo la acción correctamente.
Requisitos relacionados	RSF-08

TABLA 5.13 PS-09

PS-10	
Nombre	Detención mediante control por acciones

PS-10	
Descripción	El usuario puede ejecutar una detención del robot utilizando el control por acciones. Para ello, el usuario podrá pulsar el botón de la acción correspondiente.
Precondiciones	1. Abrir la aplicación 2. Conexión con el servidor 3. Selección de modo de control por acciones
Criterios de aceptación	La aplicación manda correctamente el ID de la acción y el robot lleva a cabo la acción correctamente.
Requisitos relacionados	RSF-08

TABLA 5.14 PS-10

PS-11	
Nombre	Giro continuo a la izquierda mediante control por acciones
Descripción	El usuario puede ejecutar un giro continuo a la izquierda utilizando el control por acciones. Para ello, el usuario podrá pulsar el botón de la acción correspondiente.
Precondiciones	1. Abrir la aplicación 2. Conexión con el servidor 3. Selección de modo de control por acciones
Criterios de aceptación	La aplicación manda correctamente el ID de la acción y el robot lleva a cabo la acción correctamente.
Requisitos relacionados	RSF-08

TABLA 5.15 PS-11

PS-12	
Nombre	Giro continuo a la derecha mediante control por acciones
Descripción	El usuario puede ejecutar un giro continuo a la derecha utilizando el control por acciones. Para ello, el usuario podrá pulsar el botón de la acción correspondiente.

PS-12	
Precondiciones	1. Abrir la aplicación 2. Conexión con el servidor 3. Selección de modo de control por acciones
Criterios de aceptación	La aplicación manda correctamente el ID de la acción y el robot lleva a cabo la acción correctamente.
Requisitos relacionados	RSF-08

TABLA 5.16 PS-12

PS-13	
Nombre	Aceleración mediante control por acciones
Descripción	El usuario puede aumentar la velocidad del robot utilizando el control por acciones. Para ello, el usuario podrá pulsar el botón de la acción correspondiente.
Precondiciones	1. Abrir la aplicación 2. Conexión con el servidor 3. Selección de modo de control por acciones
Criterios de aceptación	La aplicación manda correctamente el ID de la acción y el robot lleva a cabo la acción correctamente.
Requisitos relacionados	RSF-08

TABLA 5.17 PS-13

PS-14	
Nombre	Freno mediante control por acciones
Descripción	El usuario puede disminuir la velocidad del robot utilizando el control por acciones. Para ello, el usuario podrá pulsar el botón de la acción correspondiente.
Precondiciones	1. Abrir la aplicación 2. Conexión con el servidor 3. Selección de modo de control por acciones

PS-14	
Criterios de aceptación	La aplicación manda correctamente el ID de la acción y el robot lleva a cabo la acción correctamente.
Requisitos relacionados	RSF-08

TABLA 5.18 PS-14

PS-15	
Nombre	Movimiento trasero continuo mediante control por acciones
Descripción	El usuario puede ejecutar un movimiento trasero continuo utilizando el control por acciones. Para ello, el usuario podrá pulsar el botón de la acción correspondiente.
Precondiciones	1. Abrir la aplicación 2. Conexión con el servidor 3. Selección de modo de control por acciones
Criterios de aceptación	La aplicación manda correctamente el ID de la acción y el robot lleva a cabo la acción correctamente.
Requisitos relacionados	RSF-08

TABLA 5.19 PS-15

PS-16	
Nombre	Movimiento frontal con distancia mediante control por acciones
Descripción	El usuario puede ejecutar un movimiento frontal con distancia utilizando el control por acciones. Para ello, el usuario deberá pulsar el botón de la acción correspondiente y, posteriormente, definir una distancia a recorrer.
Precondiciones	1. Abrir la aplicación 2. Conexión con el servidor 3. Selección de modo de control por acciones 4. Selección de acción

PS-16	
Criterios de aceptación	La aplicación manda correctamente el ID de la acción y los parámetros necesarios al servidor, y el robot lleva a cabo la acción correctamente.
Requisitos relacionados	RSF-08, RSF-09

TABLA 5.20 PS-16

PS-17	
Nombre	Movimiento trasero con distancia mediante control por acciones
Descripción	El usuario puede ejecutar un movimiento trasero con distancia utilizando el control por acciones. Para ello, el usuario deberá pulsar el botón de la acción correspondiente y, posteriormente, definir una distancia a recorrer.
Precondiciones	1. Abrir la aplicación 2. Conexión con el servidor 3. Selección de modo de control por acciones 4. Selección de acción
Criterios de aceptación	La aplicación manda correctamente el ID de la acción y los parámetros necesarios al servidor, y el robot lleva a cabo la acción correctamente.
Requisitos relacionados	RSF-08, RSF-10

TABLA 5.21 PS-17

PS-18	
Nombre	Movimiento continuo con ángulo mediante control por acciones
Descripción	El usuario puede ejecutar un movimiento continuo con ángulo utilizando el control por acciones. Para ello, el usuario deberá pulsar el botón de la acción correspondiente y, posteriormente, definir un ángulo para girar.

PS-18	
Precondiciones	1. Abrir la aplicación 2. Conexión con el servidor 3. Selección de modo de control por acciones 4. Selección de acción
Criterios de aceptación	La aplicación manda correctamente el ID de la acción y los parámetros necesarios al servidor, y el robot lleva a cabo la acción correctamente.
Requisitos relacionados	RSF-08, RSF-11

TABLA 5.22 PS-18

PS-19	
Nombre	Movimiento con distancia y ángulo mediante control por acciones
Descripción	El usuario puede ejecutar un movimiento con distancia y ángulo utilizando el control por acciones. Para ello, el usuario deberá pulsar el botón de la acción correspondiente y, posteriormente, definir una distancia y ángulo a recorrer.
Precondiciones	1. Abrir la aplicación 2. Conexión con el servidor 3. Selección de modo de control por acciones 4. Selección de acción
Criterios de aceptación	La aplicación manda correctamente el ID de la acción y los parámetros necesarios al servidor, y el robot lleva a cabo la acción correctamente.
Requisitos relacionados	RSF-08, RSF-12

TABLA 5.23 PS-19

PS-20	
Nombre	Giro con grados mediante control por acciones

PS-20	
Descripción	El usuario puede ejecutar un giro con ángulo utilizando el control por acciones. Para ello, el usuario deberá pulsar el botón de la acción correspondiente y, posteriormente, definir un ángulo para girar.
Precondiciones	1. Abrir la aplicación 2. Conexión con el servidor 3. Selección de modo de control por acciones 4. Selección de acción
Criterios de aceptación	La aplicación manda correctamente el ID de la acción y los parámetros necesarios al servidor, y el robot lleva a cabo la acción correctamente.
Requisitos relacionados	RSF-08, RSF-13

TABLA 5.24 PS-20

PS-21	
Nombre	Movimiento por coordenadas mediante control por acciones
Descripción	El usuario puede ejecutar un movimiento por coordenadas utilizando el control por acciones. Para ello, el usuario deberá pulsar el botón de la acción correspondiente y, posteriormente, definir las coordenadas a las que trasladar el robot.
Precondiciones	1. Abrir la aplicación 2. Conexión con el servidor 3. Selección de modo de control por acciones 4. Selección de acción
Criterios de aceptación	La aplicación manda correctamente el ID de la acción y los parámetros necesarios al servidor, y el robot lleva a cabo la acción correctamente.
Requisitos relacionados	RSF-08, RSF-14

TABLA 5.25 PS-21

5.2.2. Pruebas de implantación del sistema

Las pruebas de implantación tienen como objetivo asegurar que todos los procedimientos y componentes funcionan correctamente en el entorno en el que se vayan a implementar. Es necesario establecer estas pruebas para que ajusten el sistema a los requerimientos demandados por el entorno operacional.

Las especificaciones de las pruebas de Implementación serán reflejadas en el siguiente modelo de tabla:

Identificador	
Objetivo	
Ámbito	
Procedimiento	
Aceptación	

TABLA 5.26 FORMATO DE TABLA DE PRUEBAS DE IMPLEMENTACIÓN

Donde:

- **Identificador**: Cada prueba tendrá un identificador único. Este tendrá el formato "PIM-XX", donde "PIM" hace referencia a "Prueba de Implantación" y "XX" será un número identificativo de dos dígitos.
- **Objetivo**: breve resumen de la función de la prueba.
- **Ámbito**: Característica de implantación analizada por la prueba (Rendimiento, Seguridad, Requisitos del sistema operativo…)
- **Procedimiento**: Fases por la que pasará la prueba.
- **Aceptación**: criterios de aceptación de la prueba.

PIM-01	
Objetivo	Asegurar que el sistema funciona correctamente con la conexión Wifi del dispositivo.
Ámbito	Rendimiento del sistema con el dispositivo

PIM-01	
Procedimiento	1. Activar la conexión a internet (Wifi) 2. Iniciar la aplicación 3. Navegar por las distintas opciones de la aplicación. 4. Analizar el rendimiento
Aceptación	El sistema debe responder correctamente a las solicitudes online del usuario.

TABLA 5.27 PIM-01

PIM-02	
Objetivo	Asegurar que el sistema se recupera de manera apropiada tras sufrir un fallo.
Ámbito	Recuperación
Procedimiento	1. Simular un fallo en el sistema 2. Observar cómo responde la aplicación
Aceptación	En caso de fallo, la aplicación deberá mostrar un mensaje de error y llevar a cabo el procedimiento correspondiente dependiendo del tipo de error.

TABLA 5.28 PIM-02

PIM-03	
Objetivo	Asegurar que el servidor está operativo de forma constante tras iniciarlo.
Ámbito	Requisito del sistema
Procedimiento	1. Realizar conexión desde la aplicación al servidor. 2. Comprobar que la aplicación se comunica correctamente con el servidor.
Aceptación	El servidor debe funcionar en todo momento, siendo capaz de responder a las peticiones de sistema correctamente.

TABLA 5.29 PIM-03

5.2.3. Posibilidades de las pruebas del sistema

En este apartado se han desarrollado todas las posibles acciones que puede realizar un usuario para cada prueba del sistema en la que exista más de una posibilidad, para así cubrir cada caso posible del sistema. Para la representación de las posibilidades de acciones se utilizará el siguiente formato de tabla:

ID	Nombre	Descripción	Resultado

TABLA 5.30 FORMATO DE TABLA DE POSIBILIDADES DE LAS PRUEBAS

Donde:

- **ID**: Identificador de la prueba.
- **Nombre**: Nombre de la prueba.
- **Descripción**: Descripción de acciones realizadas para obtener un resultado para la prueba.
- **Resultado**: Resultado que se obtiene al realizar las acciones de la prueba.

ID	Nombre	Descripción	Resultado
PS-01	Conexión con el servidor mediante conexión Wifi	El usuario rellena los campos de dirección IP y Puerto correctamente.	Se realiza la conexión con el servidor y se transita a la actividad de elección de modo.
PS-01	Conexión con el servidor mediante conexión Wifi	El usuario rellena uno o ambos campos de forma incorrecta.	No se realiza la conexión y el sistema muestra un mensaje de error.
PS-01	Conexión con el servidor mediante conexión Wifi	El usuario deja uno o ambos campos sin rellenar.	No se realiza la conexión y el sistema muestra un mensaje de error.
PS-16	Movimiento frontal con distancia mediante control por acciones	El usuario rellena el campo de distancia correctamente.	Se envían los datos al servidor y se ejecuta la acción en el robot.

ID	Nombre	Descripción	Resultado
PS-16	Movimiento frontal con distancia mediante control por acciones	El usuario rellena el campo de forma incorrecta.	No se envían los datos y se muestra un mensaje de error en la aplicación.
PS-16	Movimiento frontal con distancia mediante control por acciones	El usuario deja el campo sin rellenar.	No se envían los datos y se muestra un mensaje de error en la aplicación.
PS-17	Movimiento trasero con distancia mediante control por acciones	El usuario rellena el campo de distancia correctamente.	Se envían los datos al servidor y se ejecuta la acción en el robot.
PS-17	Movimiento trasero con distancia mediante control por acciones	El usuario rellena el campo de forma incorrecta.	No se envían los datos y se muestra un mensaje de error en la aplicación.
PS-17	Movimiento trasero con distancia mediante control por acciones	El usuario deja el campo sin rellenar.	No se envían los datos y se muestra un mensaje de error en la aplicación.
PS-18	Movimiento continuo con ángulo mediante control por acciones	El usuario rellena el campo de ángulo correctamente.	Se envían los datos al servidor y se ejecuta la acción en el robot.
PS-18	Movimiento continuo con ángulo mediante control por acciones	El usuario rellena el campo de forma incorrecta.	No se envían los datos y se muestra un mensaje de error en la aplicación.
PS-18	Movimiento continuo con ángulo mediante control por acciones	El usuario deja el campo sin rellenar.	No se envían los datos y se muestra un mensaje de error en la aplicación.
PS-19	Movimiento con distancia y ángulo mediante control por acciones	El usuario rellena los campos de distancia y ángulo de forma correcta.	Se envían los datos al servidor y se ejecuta la acción en el robot.

ID	Nombre	Descripción	Resultado
PS-19	Movimiento con distancia y ángulo mediante control por acciones	El usuario rellena uno o ambos campos de forma incorrecta.	No se envían los datos y se muestra un mensaje de error en la aplicación.
PS-19	Movimiento con distancia y ángulo mediante control por acciones	El usuario deja uno o ambos campos sin rellenar.	No se envían los datos y se muestra un mensaje de error en la aplicación.
PS-20	Giro con grados mediante control por acciones	El usuario rellena el campo de ángulo correctamente.	Se envían los datos al servidor y se ejecuta la acción en el robot.
PS-20	Giro con grados mediante control por acciones	El usuario rellena el campo de forma incorrecta.	No se envían los datos y se muestra un mensaje de error en la aplicación.
PS-20	Giro con grados mediante control por acciones	El usuario deja el campo sin rellenar.	No se envían los datos y se muestra un mensaje de error en la aplicación.
PS-21	Movimiento por coordenadas mediante control por acciones	El usuario rellena los campos de las coordenadas de forma correcta.	Se envían los datos al servidor y se ejecuta la acción en el robot.
PS-21	Movimiento por coordenadas mediante control por acciones	El usuario rellena uno o ambos campos de forma incorrecta.	No se envían los datos y se muestra un mensaje de error en la aplicación.
PS-21	Movimiento por coordenadas mediante control por acciones	El usuario deja uno o ambos campos sin rellenar.	No se envían los datos y se muestra un mensaje de error en la aplicación.

TABLA 5.31 POSIBILIDADES DE LAS PRUEBAS DEL SISTEMA

6. PLANIFICACIÓN DEL PROYECTO

En este apartado se desarrolla todo lo relacionado con la planificación del proyecto, la organización y el modelo de desarrollo llevados a cabo durante la elaboración, así como un desarrollo de la planificación.

6.1.Organización del desarrollo

Debido a la costumbre del uso del modelo en cascada y de su facilidad de utilización en el desarrollo de proyectos de software, este ha sido el modelo escogido para este proyecto. Consiste en la visualización del proyecto en diferentes fases, uniendo los pasos que se siguen hacia abajo, estableciendo una similitud con una cascada. Siguiendo las bases de este tipo de modelo, se ha controlado todo el desarrollo del proyecto mediante la elaboración de la documentación establecida en el presente documento, y se han tenido en cuenta todos los plazos de comienzo y fin de las diferentes tareas que componen el proyecto.

Las fases establecidas para el proyecto en base al modelo en cascada son las siguientes:

- **Investigación inicial**: Primera fase del desarrollo, en la cual se ha buscado documentación relevante para la elaboración del proyecto, posibilidad de la existencia de sistemas similares y estudio de las alternativas de desarrollo.
- **Análisis de requisitos**: Tras recopilar toda la información necesaria acerca de la situación previa al desarrollo, se llevó a cabo un análisis de los requisitos del cliente con respecto al sistema que requería. Se propusieron diferentes alternativas de desarrollo, así como diferentes funcionalidades para completar el sistema.
- **Diseño del sistema**: En esta fase, tras obtener un análisis de requisitos completo, se realizó con el objetivo de tener una visión clara y eficiente de cómo aplicar y cumplimentar todos los requisitos establecidos en la fase anterior. Se escogieron las alternativas para el desarrollo del sistema y se establecieron las pautas del desarrollo.
- **Codificación y montaje**: Fase principal de la creación del sistema, en la cual se lleva a cabo lo desarrollado durante el análisis y el diseño del sistema a la realidad. En esta fase, se desarrollaron tanto la aplicación móvil como el servidor, además del montaje de la placa en el robot y la instalación de las

dependencias necesarias en la placa para la posterior instalación del servidor en ésta.

- **Realización de pruebas**: Tras llevar a cabo el desarrollo de la aplicación y el servidor, así como el montaje del sistema, se llevaron a cabo las diferentes pruebas del sistema para cumplimentar que éste funciona de forma correcta y eficiente, con el objetivo de demostrar el correcto desarrollo todas las necesidades del análisis y del diseño.
- **Verificación del sistema**: Por último, se llevó a cabo una verificación del sistema por parte del cliente, mediante la cual éste pudo tener una toma de contacto con el sistema y probar que todos los requisitos demandados se encontraban cubiertos por el sistema.

6.2.Planificación

Inicialmente, el proyecto se pensó para ser entregado del 21 al 25 de septiembre de 2018. El comienzo del desarrollo se produjo la semana del día 23 de julio, estableciendo un máximo de 10 semanas para desarrollar por completo el proyecto. Se estableció una media de 6 horas diarias de trabajo sin contar fines de semana. La última semana debe estar acabado el proyecto, con el objetivo de tener días de margen para correcciones. Teniendo en cuenta este valor, la cantidad de horas aproximada requerida para el desarrollo completo de proyecto es de 300 horas. En base a las fases establecidas en el apartado previo, se estableció una estimación de desarrollo del proyecto:

Estimación de duración del proyecto		
Fase del proyecto	**Fecha de comienzo**	**Fecha de fin**
Investigación inicial	23 de julio de 2018	1 de agosto de 2018
Análisis de requisitos	2 de agosto de 2018	10 de agosto de 2018
Diseño del sistema	13 de agosto de 2018	22 de agosto de 2018
Codificación y montaje	23 de agosto de 2018	6 de septiembre de 2018
Realización de pruebas	5 de septiembre de 2018	17 de septiembre de 2018
Verificación del sistema	18 de septiembre de 2018	21 de septiembre de 2018

TABLA 6.1 ESTIMACIÓN DE DURACIÓN DEL PROYECTO

El Gantt estimado de la duración del proyecto es el siguiente:

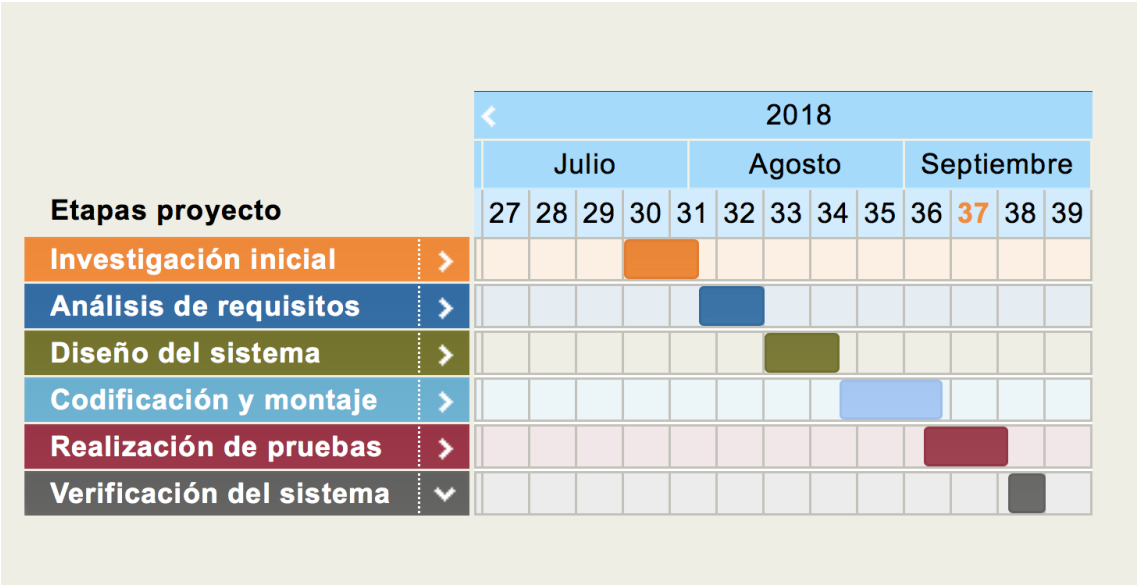

Fig. 6.1 Gantt estimado del proyecto

En el Gantt pueden observarse las semanas del año en las que se realiza el proyecto, siendo 9 el total y dejando la décima para revisiones. Mediante esta estimación, si se cumplieran los plazos, el proyecto debería ser finalizado el día 21, 4 días de margen para solución de imprevistos finales.

Tras haber finalizado el desarrollo del proyecto, los valores reales tomados a partir de las fechas marcadas tanto al comienzo como al final del desarrollo de cada fase son los siguientes:

Duración real del proyecto		
Fase del proyecto	**Fecha de comienzo**	**Fecha de fin**
Investigación inicial	23 de julio de 2018	27 de julio de 2018
Análisis de requisitos	30 de julio de 2018	7 de agosto de 2018
Diseño del sistema	8 de agosto de 2018	17 de agosto de 2018
Codificación y montaje	20 de agosto de 2018	5 de septiembre de 2018
Realización de pruebas	6 de septiembre de 2018	14 de septiembre de 2018
Verificación del sistema	17 de septiembre de 2018	21 de septiembre de 2018

TABLA 6.2 DURACIÓN REAL DEL PROYECTO

Analizando la duración de cada fase, en comparación con la estimada correspondiente, se puede observar que la mayoría de las fases finalizan antes de lo previsto, dando pie a

que la siguiente pueda comenzarse antes. Sin embargo, debido a una mala estimación del tiempo de codificación, se puede observar como la fase de Codificación y montaje se alarga 2 días más de lo estimado inicialmente. Esto implicó que el ahorro de tiempo conseguido en las fases previas fuese útil para que el retraso en la fase de codificación no implicara un retraso general del proyecto. Tanto la realización de pruebas como la verificación del sistema se han llevado a cabo en las fechas previstas.

El Gantt final del proyecto pasa a ser el siguiente:

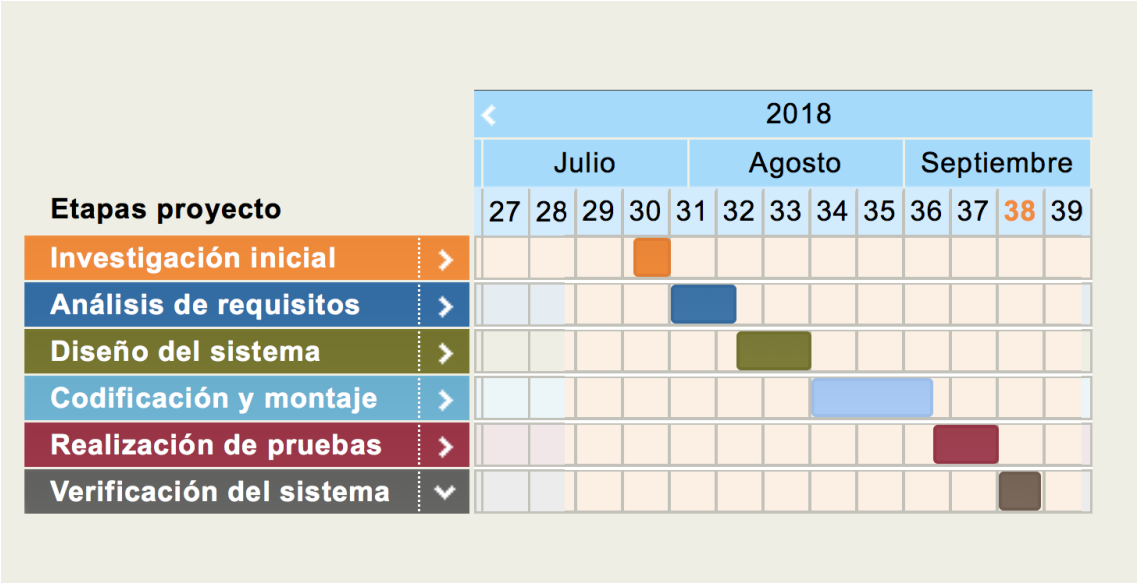

Fig. 6.2 Gantt real del proyecto

7. MARCO REGULADOR

En el presente apartado se analizan tanto la legislación atribuible al proyecto como los estándares técnicos aplicables.

7.1.Legislación

En cuanto a licencias aplicables al desarrollo del proyecto, para ARIA se aplica la GNU *General Public License* (GPL) [11]. Esta licencia de derechos de autor permite a los usuarios utilizar, copiar y modificar el software al que se aplique la licencia.

Por otro lado, se han analizado tanto el Reglamento General de Protección de datos (RGPD) [12] como la Ley Orgánica de Protección de Datos (LOPD) [13] en relación con la posibilidad de guardar datos del usuario en el sistema y la necesidad de protegerlos. En base al diseño realizado para el sistema, se decidió que no era necesario el guardado de datos sensibles de los usuarios que utilizaran el sistema. Sin embargo, para el correcto funcionamiento del sistema, ha sido necesario guardar la dirección IP y puerto en el que el servidor se ejecuta en el fichero *SharedPreferences* de la aplicación. Ya que el sistema se ha desarrollado para ser utilizado en una red local cerrada, estos datos no son útiles fuera del propio sistema y, por lo tanto, ninguna de las leyes previamente mencionadas aplica.

Por último, pese a que no es necesario aplicar ninguna de estas leyes de protección de datos, es necesario mencionar que, al ser una posibilidad la modificación de los robots y la adición de accesorios con capacidad de analizar el entorno, habría que tener en cuenta estas leyes en caso de uso de estas modificaciones.

En cuanto a los riesgos de uso del sistema, hay que aclarar que el sistema no tiene ningún método de prevención de riesgos como golpes o caídas, siendo responsabilidad del usuario la forma en la que se emplea el sistema. Se aplica el mismo principio en cuanto a las responsabilidades éticas, ya que es responsabilidad del usuario como se emplee el sistema.

7.2.Estándares técnicos

En cuanto a los estándares técnicos referentes al robot, se aplican varios en referencia a su seguridad:

- ISO 12100 [14]: Esta norma establece los peligros relevantes que un fabricante debe tener en cuenta de cara a la fabricación de un robot.
- ISO 14121-2 [15]: Esta norma está realizada en base a la anterior, y ofrece una guía para tratar los diferentes peligros y ejemplos de ello.
- ISO 13849-1 [16]: En base a la cantidad de riesgos y su nivel de peligrosidad, esta norma establece una clasificación para evaluar estas características y establecer unos pasos para poder asegurar que el robot no supone ningún peligro.

En cuanto al desarrollo del sistema, se ha aplicado el protocolo de internet IPv6 [17], que viene a sustituir al protocolo IPv4 [18], que ofrecía 2^{32} direcciones posibles y que fueron consumidas a principios de 2010. Este nuevo protocolo aporta 2^{128} direcciones posibles, siendo suficientes para poder gestionar todos los dispositivos actuales.

Además, el diseño del sistema ha seguido la normativa *Material Design* [19], desarrollada por Google con el objetivo de que fuera la nueva capa de diseño del sistema operativo Android.

8. ENTORNO SOCIOECONÓMICO

8.1.Presupuesto general del proyecto

En este apartado se desglosa el coste total que tiene el proyecto, uniendo los costes personales con los costes de materiales necesarios para el desarrollo.

8.1.1. Costes de recursos

Inicialmente, se ha desarrollado un presupuesto en el cual se ha tenido en cuenta la amortización de los meses que se han utilizado los diferentes recursos, ya que todos ellos se encontraban disponibles para su uso. Por lo tanto, en base al uso de cada recurso y de su vida útil, se ha podido establecer el coste de uso de estos recursos. Los recursos utilizados para el desarrollo del proyecto son los siguientes.

Coste amortizado de recursos físicos				
Recurso	**Uso del proyecto (meses)**	**Vida útil (meses)**	**Precio (€)**	**Coste final(€)**
Raspberry Pi 3	2,5	36	36,71 €	2,55 €
HP Pavilion 15-p245sa	2,5	36	500,00 €	34,72 €
Macbook Pro	1	72	2000,00 €	27,78 €
Pioneer 3-DX	2,5	120	4000,00 €	83,33 €
Televisor OKI	2	84	150,00 €	3,57 €
Xiaomi Redmi Note 4	2,5	24	200,00 €	20,83 €
Oneplus 5T	2,5	24	515,00 €	53,65 €
			TOTAL	**226,34** €

TABLA 8.1 COSTE AMORTIZADO DE RECURSOS FÍSICOS

Sin embargo, este presupuesto es diferente en caso de que el usuario que quiera utilizar el sistema no disponga de los componentes necesarios para ello. Los recursos necesarios, junto con su precio, es el siguiente:

Recurso	Precio (€)
Raspberry Pi 3	36,71 €

Recurso	Precio (€)
Pioneer 3-DX	4000,00 €
Total	**4036,71 €**

TABLA 8.2 COSTE INICIAL DE RECURSOS FÍSICOS NECESARIOS

Por otro lado, también hay licencias de software que se han utilizado durante el desarrollo del proyecto y que hay que tener en cuenta para el cálculo del presupuesto.

Coste de recursos de software				
Recurso	**Uso del proyecto (meses)**	**Vida útil (meses)**	**Precio (€)**	**Coste final(€)**
Microsoft Windows 10	2,5	36	145,00	10,07 €
Microsoft Office 365	2,5	Dependiente de la suscripción.	Dependiente de la suscripción.	0,00*€
MacOS	1	72	0,00	0,00*€
			TOTAL	**10,07 €**

TABLA 8.3 COSTE DE RECURSOS DE SOFTWARE

* Los costes de las licencias de Microsoft Office y MacOS tienen un coste de 0,00 € debido a que la primera es otorgada a los alumnos de la Universidad Carlos III de Madrid [2] de forma totalmente gratuita durante todo el periodo de curso del grado, y la segunda viene incluida dentro del precio del recurso MacBook Pro.

8.1.2. Costes personales

Por último, hay que tener en cuenta el coste personal del desarrollo del proyecto. Se ha establecido un precio por hora en base a la fase de desarrollo del proyecto, para así poder establecer un coste estimado del desarrollo, y un coste final en base a las horas reales trabajadas. Estos costes se han establecido en función de un estudio que evalúa el salario anual medio de trabajadores en el sector TIC [20].

Precio por fase de desarrollo	
Fase del proyecto	**Precio/hora (€/h)**
Investigación inicial	12 €/h

Precio por fase de desarrollo	
Análisis de requisitos	13 €/h
Diseño del sistema	14 €/h
Codificación y montaje	15 €/h
Realización de pruebas	13 €/h
Verificación del sistema	12 €/h

TABLA 8.4 PRECIO DE DESARROLLO POR FASE

En base al precio por hora de cada una de las fases, se ha redactado una tabla para establecer los costes estimados y los reales:

Coste personal				
Fase del proyecto	**Horas estimadas**	**Horas reales**	**Coste estimado (€)**	**Coste real (€)**
Investigación inicial	48	30	576,00 €	360,00 €
Análisis de requisitos	42	42	546,00 €	546,00 €
Diseño del sistema	48	48	672,00 €	672,00 €
Codificación y montaje	66	78	990,00 €	1.170,00 €
Realización de pruebas	42	42	546,00 €	546,00 €
Verificación del sistema	24	24	288,00 €	288,00 €
		TOTAL	**3.618,00 €**	**3.582,00 €**

TABLA 8.5 COSTE PERSONAL

8.1.3. Coste total

En base a los costes de recursos y costes personales, los costes totales pueden clasificarse en costes totales de desarrollo del proyecto y costes totales para el usuario.

- **Costes totales de desarrollo del proyecto**

Para calcular el coste total del desarrollo del proyecto, es necesario tener en cuenta los costes de recursos físicos amortizados, los costes de recursos de software y los costes personales.

Costes	Precio
Costes de recursos físicos	226,43 €
Costes de recursos de software	10,07 €
Costes personales	3582,00 €
Total sin IVA	**3818,50 €**
IVA	801,89 €
Total con IVA	**4620,39 €**

TABLA 8.6 COSTE TOTAL DEL PROYECTO

- **Costes totales para el usuario**

En el caso del usuario, hay que tener en cuenta únicamente los costes iniciales del proyecto, situados en la TABLA 8.2 COSTE INICIAL DE RECURSOS FÍSICOS NECESARIOS, ya que el usuario final no tiene que pagar los costes de personal ni los costes de recursos físicos amortizados y de software.

Recurso	Precio (€)
Raspberry Pi 3	36,71 €
Pioneer 3-DX	4000,00 €
Total	**4036,71 €**

TABLA 8.7 COSTES FINALES DEL USUARIO

8.2.Impacto socioeconómico

En este apartado se desarrolla el impacto esperado del sistema final del proyecto, explorando las diferentes alternativas. Tal como se explicó en el apartado 3.1 del presente documento, existen 3 tipos de usuarios potenciales que puedan usar el sistema. Inicialmente, el desarrollo del proyecto se ha llevado a cabo con el objetivo de satisfacer los ámbitos de la educación y la enseñanza, correspondientes a los 2 primeros tipos de usuarios. Sin embargo, gracias al uso genérico del sistema, el tercer grupo de usuarios también tiene la posibilidad de utilizar el sistema sin mayor problema.

Principalmente, este sistema se presenta como una buena alternativa a la necesidad del uso de solo un programa ejecutando en la placa para manejar el robot. Ya que el sistema no se ha desarrollado con el objetivo de ser una alternativa directa a otros sistemas de

comunicación inalámbrica, éste se convierte en la primera alternativa a poder manejar los diferentes robots mediante un dispositivo externo que los controle.

Por otro lado, una de las ventajas del sistema desarrollado es que es fácilmente modificable por dos motivos:

- La aplicación móvil funciona mediante actividades, las cuales pueden ser ampliadas en cualquier momento con la adición de una línea de código.
- El servidor, al ser dependiente de la librería ARIA [1], puede adaptarse sin demasiados problemas a trabajos previos en los que se hayan desarrollado ya algunos movimientos concretos para algún robot.

Estas ventajas hacen que las posibilidades del sistema sean enormes. Cualquier usuario, si lo requiere, podría añadir nuevas funcionalidades que ayuden en cualquiera de las labores que los diferentes tipos de usuarios puedan desempeñar, desde investigación hasta trabajo de campo.

Por otro lado, el sistema puede ser útil para más tipos de usuarios aparte de los analizados previamente. Gracias a la capacidad de cargar diferentes cargas de peso de los robots, puede servir para muchas otras tareas. Por ejemplo, traslado de materiales de construcción pesados o transporte de diferentes objetos para personas con movilidad reducida.

Además de todo esto, gracias a la fácil modificación de los robots, con la adición de diferentes accesorios, puede ampliar aún más la usabilidad del sistema, como por ejemplo la visualización en tiempo real del entorno del robot.

Sin embargo, también hay que valorar algún impacto negativo. Este sistema ha sido desarrollado gracias a la disponibilidad de uno de los robots en el Departamento de Informática de la Universidad Carlos III de Madrid [2]. Pero, de cara al nuevo usuario, la necesidad de hacer un alto desembolso para poder utilizar el sistema podría provocar que el uso del sistema no sea todo lo bueno que se espera.

Por lo tanto, finalizando, el sistema se muestra como una opción completa y polivalente, a la vez que ampliable a diferentes entornos y posibilidades, para manejar cualquier tipo de robot compatible con la librería ARIA. Además, con el objetivo de ampliar la difusión del proyecto, éste será publicado en el repositorio de la Universidad Carlos III

de Madrid. Gracias a los diferentes tipos de robot, y la adaptabilidad del sistema, las posibilidades de utilización y desarrollo son muy altas, sirviendo para diversas situaciones.

9. CONCLUSIONES

9.1.Conclusiones y objetivos cumplidos

Como se estableció en el apartado 1.3 del presente documento, existen diferentes objetivos que se ha buscado cumplir durante el desarrollo del proyecto.

El primero de estos objetivos se centró en el desarrollo del servidor, que se ejecuta en la placa controladora del robot. Este objetivo fue el primero en ser cumplido, ya que la librería de ARIA [1] brinda diferentes programas con el objetivo de poder funcionar como cliente ante un servidor. Se escogió realizar el servidor mediante sockets en vez de utilizar ArNetworking ante la facilidad que la primera opción ofrecía en cuanto a funcionamiento, implementación y compatibilidad con el lenguaje Java. Así, el servidor permite, gracias a ARIA, ejecutar diferentes acciones sobre el robot y modificar sus atributos para poder satisfacer los objetivos del usuario.

El segundo de los objetivos fue realizar un cliente en lenguaje Java con independencia de la plataforma, que pudiera conectarse con el servidor y mandar los datos necesarios a éste para conseguir una respuesta en el robot. Este objetivo, gracias a los diferentes ejemplos aportados por la librería de ARIA, fue más sencillo de realizar que el servidor. Se realizaron pruebas tanto en la propia placa del servidor como desde otro computador separado.

El tercer objetivo se basó en desarrollar la aplicación Android y trasladar las funcionalidades del cliente al sistema de la aplicación. Gracias a la experiencia obtenida en el último curso del grado, el desarrollo de la aplicación fue bastante más sencillo gracias a la práctica en este entorno. Aun así, ciertas partes llevaron más tiempo del esperado, haciendo que la fase de codificación se alargara un poco. Las pruebas mostradas en el apartado 5 del presente documento se realizaron tanto en el cliente del segundo objetivo, obteniendo los resultados esperados y, tras realizar la aplicación, también se probó en ésta, obteniendo los mismos resultados que con el cliente anterior.

Por lo tanto, los objetivos pretendidos al comienzo del proyecto se han cumplido.

9.2.Futuras líneas de trabajo

Además de los objetivos iniciales, también se propuso en reuniones con el cliente el desarrollar un planificador automático con el objetivo de que el robot pudiera auto

manejarse. En base a la duración del proyecto, esta idea se desestimó, pero podría ser una buena línea futura de investigación para la mejora del sistema.

Por otro lado, la funcionalidad de movimiento por coordenadas ha sido implementada de forma simple, con el objetivo de ser eficaz. Sin embargo, la existencia de obstáculos podría dificultar o imposibilitar llegar al objetivo marcado. Por lo tanto, añadir al sistema la capacidad de esquivar obstáculos y calcular la mejor ruta posible sería otra línea futura de trabajo que mejoraría mucho el sistema.

Además, hay que tener en cuenta la capacidad de mejorar al robot mediante la adición de accesorios, lo cual ofrece muchas funcionalidades extra, como una mejor detección del entorno o visibilidad en tiempo real mediante una cámara, lo cual podría ayudar al usuario a manejar el robot. También podría ser de mucha utilidad la adición de algún tipo de indicador para mostrar la dirección IP y el puerto, como por ejemplo una pantalla pequeña colocada sobre la placa.

10. CONTROL OF A ROBOT WITH RASPBERRY PI

10.1. Introduction and objetives

Nowadays, the robot sector is very widespread. Over the last few years, mobile robot companies, such as MobileRobots [5], have manufactured different types of robots with different objectives, such as teaching, carrying out work on irregular terrains, facilitating user interaction with robots or improve the efficiency of different tasks. In this case, this Final Degree Project focuses on mobile robots.

The main characteristic of mobile robots is the ability to move in their environment, unlike fixed industrial robots. These robots are an important point of the current research of universities with laboratories focused on robotics.

The main objective of this End of Degree Project is to manage one of the robots wirelessly, allowing it to be able to perform different movements, such as movements with distances or turns with specific degrees.

10.1.1. Context

At the Carlos III University of Madrid [2], the IT Department has several units of the Pioneer P3-DX mobile robot. This robot has been manufactured by Pioneer, in collaboration with MobileRobots [5], and initially has two wheels, each with an independent motor, producing that the robot is able to rotate on its axis, allowing the wheels to turn at different speeds and in different directions. It also has a front sonar to get a better control of the environment. In addition, this robot is easily customizable, since there are dozens of accessories tested by Pioneer for use with this robot, such as lasers for distance control, vision cameras or mechanical arms to pick up objects.

To carry out the control of this robot, it is necessary to use a computer that, connected to the robot, can send you information. With this objective, ARIA [1] is available. ARIA is an object-oriented robot control application programming interface, compatible with MobileRobots robots. ARIA is developed in C ++ and offers a high-level control over the robot, as well as the different accessories that can be added. In addition, ARIA is available for Linux, Mac and Windows, making it compatible with all current operating systems.

10.1.2. Project motivation

Currently, the control of these robots is done through client-server connections between a computer and the controller board of the robot, or only the controller board performs actions on the robot by making decisions autonomously.

The first type of control is done through a TCP connection between the client and the server. However, this connection, which is made through the ArNetworking library, included in ARIA, is developed in C ++, hindering the development of mobile platforms that use this type of connection using this library.

The second type of control is performed only on the board connected to the robot by making decisions with artificial intelligence. However, this type of control is not useful for controlling robots manually.

Currently, there is no way to control this type of robots manually by using smartphones, both Android and iOS. According to the web portal We Are Social [3] in its study Digital in 2018 [4], there are more than 5,000 million mobile lines active in 2018. As these reports explain, smartphones have a penetration rate in society of 68 %. This means that, for smartphone users, it is more comfortable and attractive the use of a mobile application, rather than the use of a desktop computer. Therefore, the development of a control system for this type of robot in smartphones can be very useful. This problem is the main reason why this TFG has been carried out.

10.1.3. Objetives

Having analyzed the problems found in a first study of the petition, the objective of this TFG is presented as the development of a mobile application that, using a TCP connection, can connect to a board attached to the robot, being able to carry out a manual control over it, in addition to endowing him with certain useful actions of movement.

This main objective is divided into different sub-objectives, which are summarized below:

- Develop an executable server on the board connected to the robot, which performs the actions required on it.
- Develop a client capable of sending the required actions to the server.

- Develop a mobile application that implements the client, in order to send these actions from the mobile terminal.

In addition, this project has several utilities after its development:

- Provide students or future researchers with simple control over the robot.
- Facilitate teaching on the control of these robots, showing a simple and practical way to control it.

10.2. System capabilities

This system is based on a mobile application that, through a wireless connection, connects to a board placed on the robot, and through the use of commands the application will perform different actions on the robot. The robot board acts as a server, while the mobile application acts as a client. Therefore, each one of them has different functionalities that must be covered for the correct functioning of the system.

The main idea that has been used for the development of the system has been to give the user the possibility to both handle the robot manually, and to select different actions that the user wants the robot to execute. Therefore, the main functionalities of the mobile application are the following:

- Connection via wireless communication with the controller board.
- Manual control of the robot.
- Control by robot actions.

The first of the functionalities will be carried out through Wifi connection.

As for the manual control, it must allow the user to move the robot in all the possible directions it allows.

On the other hand, through the control by actions, the user must be able to execute different actions, such as continuous movements, with specific distances, speed variations or movement by coordinates.

These functionalities have been grouped in the following table, and are identified with the FUA-XX code, where FUA stands for Application Functionality, and XX is a two-digit identifier number.

Functionalities of the mobile application	
FUA-01	Connection to the server via Wi-Fi connection.
FUA-02	Manual control: Front movement.
FUA-03	Manual control: Rear movement.
FUA-04	Manual control: Turn to the right.
FUA-05	Manual control: Turn left.
FUA-06	Control per action: Continuous frontal movement.

Functionalities of the mobile application	
FUA-07	Control per action: Detention.
FUA-08	Control per action: Continuous turn to the left.
FUA-09	Control per action: Continuous turn to the right.
FUA-10	Control per action: Acceleration.
FUA-11	Control per action: Brake.
FUA-12	Control per action: Continuous rear movement.
FUA-13	Control per action: Front movement with distance.
FUA-14	Control per action: Rear movement with distance.
FUA-15	Control per action: Continuous movement with angle.
FUA-16	Control by action: Movement with distance and angle.
FUA-17	Control per action: Turn with degrees.
FUA-18	Control per action: Movement by coordinates.

TABLA 10.1 FUNCTIONALITIES OF THE MOBILE APPLICATIONS

10.3. State of the question

This chapter shows how the different sections relevant to the development of the project were before carrying it out, and with the different alternatives of each section.

10.3.1. Server

In order to exercise control over the robot, the ARIA [1] robotic control library has been used.

ARIA is the native control interface of the robots of MobileRobots [5]. It is an object-oriented robot control application programming interface. It offers manipulation on the robot both at a high level and at a low level. It is developed in C ++, but offers two wrappers to use Java or Python language for the development of both the server and the client. These wrappers are classes elaborated in the language that will be used, which can invoke the methods of the original language and interpret the results obtained from their functions. This feature has been essential in the development of the project, since the objective of making a mobile application with which to control the robot would not have been possible without this.

In addition to the corresponding wrappers, ARIA provides several examples for both Java and Python, as well as many more examples developed in C ++. Despite having the examples written in Java, these are very simple, not showing well the operation of the robot. To observe it better, it is necessary to consult both the documentation and the examples made in C ++, since these are much more complete and contribute more.

On the other hand, ARIA offers support for other functionalities through the addition of other libraries, such as distance control by sonar or laser.

At first, it is possible to test the examples provided in ARIA about a simulator developed by MobileRobots, called MobileSim [6]. This software serves as a robot simulator for MobileRobots and its environments, with the aim of developing the robot's controller software without having to have it on hand. MobileSim uses data from scenarios with obstacles, which are developed using external software such as Mapper3 [8] or Mapper3Basic [9]. These environments can be developed using the robot using laser sensors, or manually.

In addition, with the objective of monitoring and controlling the robot remotely, MobileRobots has also developed MobileEyes [7], a graphical application that uses the ArNetworking library to connect to the server. Depending on the specifications provided by the server, MobileEyes is able to represent the position of the robot on the stage.

Finally, MobileEyes offers controls to manage the robot remotely, change parameters of the robot, or send a goal for it to move.

10.3.2. Application

Since the mobile terminal market is one of the most important currently in relation to information technology, these devices are the best option to develop the client. Among the possible options for developing the application, native, pseudonative and hybrid applications have been evaluated, choosing the first option as the solution for development, specifically for the Android operating system.

The native applications are those that are developed specifically for an operating system, being necessary to adapt the programming language in case of wanting to change the target operating system. One of the main advantages of the development of native applications is the existence of very advanced IDEs at present, which makes the development and debugging easier.

On Google's mobile OS, Android, native applications are coded using Java or Kotlin as programming languages. Google also provides a very popular IDE, Android Studio. Finally, publishing an app on their store has zero cost.

There are multiple advantages of this type of applications:

- Native applications allow the use of advanced functions that each platform has.
- The performance of these applications is superior to the rest of alternatives, since they use their own native interface.
- They work independently of the availability of internet connection in the device in case the functionality of the application does not require it.
- Possibility of doing work in the background.
- Implementation of more complete and high security levels than in the rest of the alternatives.

- They improve the battery life of the terminal, since the code is optimized for the architecture used by the device.

10.3.3. Robot and controller board

During the beginning of the project, the different alternatives of robots and boards compatible with the development of the system were evaluated despite the availability of a Pioneer 3-DX robot and a reduced Raspberry Pi 3 board.

In the case of the robot, this was one of the most standardized models, with sufficient characteristics to be valid for the project. In addition, the high price of these robots made it not feasible to obtain another for the job.

On the other hand, on the side of the boards, various alternatives were evaluated to evaluate whether the Raspberry Pi 3 was suitable for the project, or else a board change was necessary. After analyzing the options, looking for a good tradeoff between technical qualities and price, the decision was to use the initial alternative as the definitive one to carry out the project.

10.4. Results

To evaluate the final performance of the project, a set of tests was carried out to demonstrate that the system worked correctly. In addition, to evaluate also the fulfillment of the delivery times, a study of the project planning was carried out.

10.4.1. System tests

To evaluate the results of the system, different tests were developed to verify the behavior of both the application and the server and the robot in the execution of each of the functionalities that the user required. These tests were run first in the ARIA emulator, MobileSim [6], collecting positive results.

After the mounting of the board and the server in the robot, the same tests as those previously used were carried out. The results are those shown in section 5 of this document, these being satisfactory and fulfilling the functionalities established by the client at the beginning of the project.

10.4.2. Project planning

With respect to compliance with the work delivery periods, an estimate of the time it would take to develop the project, as well as the final delivery date, set for September 21, leaving 4 days of margin for corrections was made at the beginning of the project.

During the development, data were taken from the start and end times of each phase, and after the completion of the project, it was possible to verify that all the phases prior to the coding completed their development earlier than expected. The coding phase was delayed two days, and its subsequent phases ended in the expected times. Therefore, the project finished on the scheduled day.

10.5. Conclusions

10.5.1. Conclussions and achieved goals

As established in section 1.3 of this document, there are different objectives that have been sought during the development of the project.

The first of these objectives focused on the development of the server, which runs on the controller board of the robot. This objective was the first to be fulfilled, since the ARIA library [1] provides different programs to be able to function as a client for the server. It was chosen to perform the server through sockets instead of using ArNetworking because the ease that the first option offered in terms of operation, implementation and compatibility with the Java language. Thus, the server allows, thanks to ARIA, to execute different actions on the robot and modify its attributes to satisfy the user's objectives.

The second objective was to make a client in Java language independently of the platform, which could connect to the server and send the necessary data to it to get an answer in the robot. This objective, thanks to the different examples provided by the ARIA library, was easier to perform than the server. Tests were performed both on the server's own board and from another separate computer.

The third objective was based on developing the Android application and transferring the client's functionalities to the application's system. Thanks to the experience obtained in the last year of the degree, the development of the application was much easier thanks to the practice in this environment. Even so, certain parts took longer than expected, making the coding phase a little longer. The tests shown in section 5 of this document were carried out both in the client of the second objective, obtaining the expected results and, after making the application, it was also tested in it, obtaining the same results as with the previous client.

Therefore, the objectives intended at the beginning of the project have been met.

10.5.2. Future lines of work

In addition to the initial objectives, it was also proposed in meetings with the client to develop an automatic planner with the objective that the robot could self-manage. Based

on the duration of the project, this idea was dismissed, but it could be a good future line of research for the improvement of the system.

On the other hand, the function of movement by coordinates has been implemented in a simple way, to be effective. However, the existence of obstacles could make it difficult or impossible to reach the marked objective. Therefore, adding to the system the ability to dodge obstacles and calculate the best possible route would be another future line of work that would greatly improve the system.

In addition, we must consider the ability to improve the robot by adding accessories, which offers many extra features, such as better detection of the environment or visibility in real time through a camera, which could help the user to manage the robot.

11.BIBLIOGRAFÍA Y ACRÓNIMOS

11.1. Referencias

- [1] MobileRobots. ARIA [Online]. Disponible en: http://robots.mobilerobots.com/wiki/ARIA, [Sep. 16, 2018].
- [2] Universidad Carlos III de Madrid [Online]. Disponible en: https://www.uc3m.es/, [Sep. 15, 2018].
- [3] *We Are Social* [Online]. Disponible en: https://wearesocial.com/, [Sep. 16, 2018].
- [4] *We Are Social*. "Digital in 2018" [Online]. Disponible en: https://wearesocial.com/blog/2018/01/global-digital-report-2018, [Sep. 16, 2018].
- [5] MobileRobots [Online]. Disponible en: http://robots.mobilerobots.com, [Sep. 16, 2018].
- [6] MobileRobots. MobileSim [Online]. Disponible en: http://robots.mobilerobots.com/wiki/MobileSim, [Sep. 16, 2018].
- [7] MobileRobots. MobileEye [Online]. Disponible en: http://robots.mobilerobots.com/wiki/MobileEyes, [Sep. 16, 2018].
- [8] MobileRobots. Mapper3 [Online]. Disponible en: http://robots.mobilerobots.com/wiki/Mapper3, [Sep. 16, 2018].
- [9] MobileRobots. Mapper3Basic [Online]. Disponible en: http://robots.mobilerobots.com/wiki/Mapper3Basic, [Sep. 16, 2018].
- [10] Open Source Robotics Foundation. Robot Operating System [Online]. Disponible en: http://wiki.ros.org/, [Sep. 16, 2018].
- [11] GNU. General Public License [Online]. Disponible en: https://www.gnu.org/licenses/gpl.html, [Sep. 17, 2018].
- [12] Reglamento del Parlamento Europeo y del Consejo de 27 de abril de 2016. Reglamento General de Protección de Datos. UE 2016/679.
- [13] Boletín Oficial del Estado de 13 de diciembre de 1999. Ley Orgánica de Protección de Datos. BOE núm.298.
- [14] "Seguridad de las máquinas. Principios generales para el diseño. Evaluación del riesgo y reducción del riesgo"; EN ISO 12100, mayo 24, 2012. [Online]. Disponible en: https://www.pilz.com/es-ES/knowhow/law-standards-norms/iso-standards/mechanic-construction/en-iso-12100, [Sep. 18, 2018].

- [15] “Safety of machinery. Risk assessment Part 2: Practical guidance and examples of methods”; ISO/TR 14121-2, junio, 2012. [Online]. Disponible en: https://www.iso.org/standard/57180.html, [Sep. 18, 2018].
- [16] “Seguridad de las máquinas. Partes de los sistemas de mando relativas a la seguridad”; EN ISO 13849-1, diciembre, 2015. [Online]. Disponible en: https://www.pilz.com/es-ES/knowhow/law-standards-norms/functional-safety/en-iso-13849-1, [Sep. 18, 2018].
- [17] S. Deering and R. Hinden, “Internet Protocol, Version 6 (IPv6)”; RFC 8200, julio, 2017. [Online]. Disponible en: https://tools.ietf.org/html/rfc8200, [Sep. 18, 2018].
- [18] “Internet protocol”; RFC 791, septiembre, 1981. [Online]. Disponible en: https://tools.ietf.org/html/rfc791, [Sep. 18, 2018].
- [19] Google, *Material Design*, [Online]. Disponible en: https://material.io/design/, [Sep. 18, 2018].
- [20] Statista. “Evolución anual del salario medio de los trabajadores en el sector TIC en España” [Online]. Disponible en: https://es.statista.com/estadisticas/866035/sueldos-de-empleados-en-el-sector-tic-segun-categoria-espana/, [Sep. 24, 2018].

11.2. Acrónimos

Acrónimo	Significado
API	Application Programming Interface
ARIA	Advanced Robot Interface for Applications
ARM	Advanced RISC Machine
BLE	Bluetooth Low Energy
CLI	Command-Line Interface
CPU	Central Processing Unit
GNU GPL	GNU General Public License
GPIO	General Purpose Input/Output
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HDMI	High-Definition Multimedia Interface

Acrónimo	Significado
HTML	HyperText Makup Language
IDE	Integrated Development Environment
IP	Internet Protocol
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
ISO	International Organization for Standardization
IVA	Impuesto sobre el Valor Añadido
LOPD	Ley Orgánica de Protección de Datos
PC	Personal Computer
PLG	Planning & Learning research Group
RAM	Random Access Memory
RFC	Request for Comments
RGPD	Reglamento General de Protección de Datos
ROS	Robot Operating System
SDK	Software Development Kit
SoC	System on a Chip
TCP	Transmission Control Protocol
TFG	Trabajo de Fin de Grado

TABLA 11.1 ACRÓNIMOS

12.ANEXO: MANUAL DE INSTALACIÓN DEL SISTEMA

12.1. Instalación del sistema

El sistema desarrollado consta de dos partes, servidor y aplicación móvil. Para llevar a cabo la instalación del servidor, primero es necesaria la instalación de ARIA [1]. Para ello, basta con entrar a su página web y descargar el paquete de instalación del sistema operativo correspondiente, y dependiendo de éste, llevar a cabo las acciones necesarias para su instalación.

- **Windows**

Para instalar ARIA en Windows, basta con ejecutar el fichero ejecutable Aria-2.9.3.exe descargado de la página web y seguir las instrucciones.

- **Linux y macOS**

La instalación en estos sistemas operativos es más complicada. Hay que descargar el código fuente, comprimido en un fichero .tar o .zip, descomprimirlo y ejecutar el fichero *Makefile* mediante el comando "make" dentro de la carpeta descomprimida.

Tras esto, se ejecuta el comando "sudo make install", lo cual instala ARIA en el directorio /usr/local/Aria. Por último, con el objetivo de poder utilizar el *wrapper* de Java, es necesario instalarlo mediante el comando "make java" en el directorio principal.

- **Código del servidor**

El código del servidor está contenido en el fichero "Servidor.java", y puede ser colocado en cualquier carpeta del sistema.

- **Aplicación móvil**

La aplicación controladora del sistema está contenida en el paquete "AriaController.apk", y se instala fácilmente descargándolo en el dispositivo móvil y ejecutando el paquete habiendo aceptado la instalación de aplicaciones de orígenes desconocidos.

12.2. Ejecución del sistema

Tras haber llevado a cabo la instalación de todo el sistema, la ejecución es bastante sencilla. Como primer paso, es necesaria la compilación y ejecución del servidor mediante los siguientes comandos:

1. javac -classpath /usr/local/Aria/Java:. Servidor.java
2. sudo java -Djava.library.path=/usr/local/Aria/lib/ -cp /usr/local/Aria/java:. Servidor

El primero de ellos lleva a cabo la compilación, y el segundo la ejecución. El *classpath* utilizado en ambos comandos muestra donde se encuentra el fichero “Aria.jar”, que contiene todos los ficheros de ejecución del *wrapper*.

Tras la ejecución del servidor, ya puede lanzarse la aplicación móvil, utilizando la dirección IP y el puerto del servidor para poder conectarse a éste.